



OVP Guide to Using Processor Models

Model specific information for Andes_D45

Imperas Software Limited
Imperas Buildings, North Weston
Thame, Oxfordshire, OX9 2HA, U.K.
docs@imperas.com



| | |
|----------|--|
| Author | Imperas Software Limited |
| Version | 20220722.0 |
| Filename | OVP_Model_Specific_Information_andes_riscv_D45.pdf |
| Created | 27 July 2022 |
| Status | OVP Standard Release |

Copyright Notice

Copyright (c) 2022 Imperas Software Limited. All rights reserved. This software and documentation contain information that is the property of Imperas Software Limited. The software and documentation are furnished under a license agreement and may be used or copied only in accordance with the terms of the license agreement. No part of the software and documentation may be reproduced, transmitted, or translated, in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without prior written permission of Imperas Software Limited, or as expressly provided by the license agreement.

Right to Copy Documentation

The license agreement with Imperas permits licensee to make copies of the documentation for its internal use only. Each copy shall include all copyrights, trademarks, service marks, and proprietary rights notices, if any.

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the readers responsibility to determine the applicable regulations and to comply with them.

Disclaimer

IMPERAS SOFTWARE LIMITED, AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Model Release Status

This model is released as part of OVP releases and is included in OVPworld packages. Please visit OVPworld.org.

Contents

| | | |
|----------|-------------------------------|----------|
| 1 | Overview | 1 |
| 1.1 | Description | 1 |
| 1.2 | Licensing | 1 |
| 1.3 | Extensions | 2 |
| 1.3.1 | Extensions Enabled by Default | 2 |
| 1.3.2 | Enabling Other Extensions | 2 |
| 1.3.3 | Disabling Extensions | 3 |
| 1.4 | General Features | 3 |
| 1.4.1 | mtvec CSR | 3 |
| 1.4.2 | stvec CSR | 4 |
| 1.4.3 | utvec CSR | 4 |
| 1.4.4 | Reset | 4 |
| 1.4.5 | NMI | 4 |
| 1.4.6 | WFI | 5 |
| 1.4.7 | cycle CSR | 5 |
| 1.4.8 | instret CSR | 5 |
| 1.4.9 | hpmcounter CSR | 5 |
| 1.4.10 | time CSR | 5 |
| 1.4.11 | mcycle CSR | 5 |
| 1.4.12 | minstret CSR | 6 |
| 1.4.13 | mhpmcounter CSR | 6 |
| 1.4.14 | Virtual Memory | 6 |
| 1.4.15 | Unaligned Accesses | 6 |
| 1.4.16 | PMP | 6 |
| 1.4.17 | LR/SC Granule | 7 |
| 1.5 | Compressed Extension | 7 |
| 1.6 | Floating Point Features | 7 |
| 1.7 | Privileged Architecture | 8 |
| 1.7.1 | Legacy Version 1.10 | 8 |
| 1.7.2 | Version 20190608 | 8 |
| 1.7.3 | Version 20211203 | 8 |
| 1.7.4 | Version 1.12 | 9 |
| 1.7.5 | Version master | 9 |
| 1.8 | Unprivileged Architecture | 9 |
| 1.8.1 | Legacy Version 2.2 | 9 |
| 1.8.2 | Version 20191213 | 9 |
| 1.9 | Other Extensions | 9 |

| | | |
|----------|---|-----------|
| 1.9.1 | Zmmul | 9 |
| 1.9.2 | Zicsr | 9 |
| 1.9.3 | Zifencei | 10 |
| 1.9.4 | Zicbom | 10 |
| 1.9.5 | Zicbop | 10 |
| 1.9.6 | Zicboz | 10 |
| 1.9.7 | Svnapot | 10 |
| 1.9.8 | Svpbmt | 10 |
| 1.9.9 | Svinal | 11 |
| 1.9.10 | Smstateen | 11 |
| 1.10 | CLIC | 11 |
| 1.11 | Load-Reserved/Store-Conditional Locking | 11 |
| 1.12 | Active Atomic Operation Indication | 12 |
| 1.13 | Interrupts | 12 |
| 1.14 | Debug Mode | 13 |
| 1.14.1 | Debug State Entry | 13 |
| 1.14.2 | Debug State Exit | 14 |
| 1.14.3 | Debug Registers | 14 |
| 1.14.4 | Debug Mode Execution | 14 |
| 1.14.5 | Debug Single Step | 15 |
| 1.14.6 | Debug Event Priorities | 15 |
| 1.14.7 | Debug Ports | 15 |
| 1.15 | Trigger Module | 15 |
| 1.15.1 | Trigger Module Restrictions | 15 |
| 1.15.2 | Trigger Module Parameters | 16 |
| 1.16 | Debug Mask | 16 |
| 1.17 | Integration Support | 17 |
| 1.17.1 | CSR Register External Implementation | 17 |
| 1.17.2 | LR/SC Active Address | 17 |
| 1.17.3 | Page Table Walk Introspection | 17 |
| 1.17.4 | Artifact Register “fflags_i” | 17 |
| 1.18 | Limitations | 18 |
| 1.19 | Verification | 18 |
| 1.20 | References | 19 |
| 2 | Andes-Specific Extensions | 20 |
| 2.1 | Andes-Specific Parameters | 20 |
| 2.1.1 | Parameter andesExtensions/mmsc_cfg | 21 |
| 2.1.2 | Parameter andesExtensions/micm_cfg | 21 |
| 2.1.3 | Parameter andesExtensions/mdcm_cfg | 22 |
| 2.1.4 | Parameter andesExtensions/uitb | 22 |
| 2.1.5 | Parameter andesExtensions/milmb | 22 |
| 2.1.6 | Parameter andesExtensions/milmbMask | 22 |
| 2.1.7 | Parameter andesExtensions/mdlmb | 22 |
| 2.1.8 | Parameter andesExtensions/mdlmbMask | 22 |
| 2.1.9 | Parameter andesExtensions/PMA_grain | 23 |
| 2.2 | Hardware Stack Protection | 23 |
| 2.3 | Physical Memory Attribute Unit | 23 |

| | | |
|----------|--|-----------|
| 2.4 | Performance Throttling | 23 |
| 2.5 | Andes-Enhanced Performance Monitoring | 23 |
| 2.6 | CSRs for CCTL Operations | 23 |
| 2.7 | Andes-Specific Instructions | 23 |
| 2.7.1 | Performance Extension Instructions | 24 |
| 2.7.1.1 | ADDIGP | 24 |
| 2.7.1.2 | BBC | 24 |
| 2.7.1.3 | BBS | 24 |
| 2.7.1.4 | BEQC | 24 |
| 2.7.1.5 | BNEC | 24 |
| 2.7.1.6 | BFOS | 25 |
| 2.7.1.7 | BFOZ | 25 |
| 2.7.1.8 | LEA.h | 25 |
| 2.7.1.9 | LEA.w | 25 |
| 2.7.1.10 | LEA.d | 25 |
| 2.7.1.11 | LEA.b.ze | 25 |
| 2.7.1.12 | LEA.h.ze | 26 |
| 2.7.1.13 | LEA.w.ze | 26 |
| 2.7.1.14 | LEA.d.ze | 26 |
| 2.7.1.15 | LBGP | 26 |
| 2.7.1.16 | LBUGP | 26 |
| 2.7.1.17 | LHGP | 26 |
| 2.7.1.18 | LHUGP | 27 |
| 2.7.1.19 | LWGP | 27 |
| 2.7.1.20 | LWUGP | 27 |
| 2.7.1.21 | LDGP | 27 |
| 2.7.1.22 | SBGP | 27 |
| 2.7.1.23 | SHGP | 28 |
| 2.7.1.24 | SWGp | 28 |
| 2.7.1.25 | SDGP | 28 |
| 2.7.1.26 | FFB | 28 |
| 2.7.1.27 | FFZMISM | 28 |
| 2.7.1.28 | FFMISM | 29 |
| 2.7.1.29 | FLMISM | 29 |
| 2.7.2 | CodeDense Instructions | 29 |
| 2.7.2.1 | EXEC.IT | 29 |
| 2.7.2.2 | EX9.IT | 29 |
| 2.7.3 | Half-Precision Load/Store Instructions | 29 |
| 2.7.3.1 | FLHW | 29 |
| 2.7.3.2 | FSHW | 29 |
| 2.8 | Andes Net Port Names | 30 |
| 3 | Configuration | 31 |
| 3.1 | Location | 31 |
| 3.2 | GDB Path | 31 |
| 3.3 | Semi-Host Library | 31 |
| 3.4 | Processor Endian-ness | 31 |
| 3.5 | QuantumLeap Support | 31 |

| | | |
|-----------|--|-----------|
| 3.6 | Processor ELF code | 31 |
| 4 | All Variants in this model | 32 |
| 5 | Bus Master Ports | 33 |
| 6 | Bus Slave Ports | 34 |
| 7 | Net Ports | 35 |
| 8 | FIFO Ports | 37 |
| 9 | Formal Parameters | 38 |
| 9.1 | Extension Parameters | 41 |
| 9.2 | Parameters with enumerated types | 42 |
| 9.2.1 | Parameter user_version | 42 |
| 9.2.2 | Parameter priv_version | 42 |
| 9.2.3 | Parameter compress_version | 42 |
| 9.2.4 | Parameter debug_version | 42 |
| 9.2.5 | Parameter rnmi_version | 43 |
| 9.2.6 | Parameter mstatus_fs_mode | 43 |
| 9.2.7 | Parameter debug_mode | 43 |
| 9.2.8 | Parameter lr_sc_constraint | 43 |
| 9.2.9 | Parameter amo_constraint | 43 |
| 9.2.10 | Parameter Zfinx_version | 43 |
| 9.2.11 | Parameter Zcea_version | 44 |
| 9.2.12 | Parameter Zceb_version | 44 |
| 9.2.13 | Parameter Zcee_version | 44 |
| 9.3 | Parameter values | 44 |
| 10 | Execution Modes | 49 |
| 11 | Exceptions | 50 |
| 12 | Hierarchy of the model | 52 |
| 12.1 | Level 1: Hart | 52 |
| 13 | Model Commands | 53 |
| 13.1 | Level 1: Hart | 53 |
| 13.1.1 | debugflags | 53 |
| 13.1.2 | dumpTLB | 53 |
| 13.1.2.1 | Argument description | 53 |
| 13.1.3 | getCSRIndex | 53 |
| 13.1.4 | isync | 53 |
| 13.1.5 | itrace | 54 |
| 13.1.6 | listCSRs | 54 |
| 13.1.6.1 | Argument description | 54 |
| 14 | Registers | 55 |
| 14.1 | Level 1: Hart | 55 |

| | | |
|--------|-------------------------------|----|
| 14.1.1 | Core | 55 |
| 14.1.2 | Floating_point | 56 |
| 14.1.3 | User_Control_and_Status | 56 |
| 14.1.4 | Supervisor_Control_and_Status | 57 |
| 14.1.5 | Machine_Control_and_Status | 58 |
| 14.1.6 | Integration_support | 60 |

Chapter 1

Overview

This document provides the details of an OVP Fast Processor Model variant.

OVP Fast Processor Models are written in C and provide a C API for use in C based platforms. The models also provide a native interface for use in SystemC TLM2 platforms.

The models are written using the OVP VMI API that provides a Virtual Machine Interface that defines the behavior of the processor. The VMI API makes a clear line between model and simulator allowing very good optimization and world class high speed performance. Most models are provided as a binary shared object and also as source. This allows the download and use of the model binary or the use of the source to explore and modify the model.

The models are run through an extensive QA and regression testing process and most model families are validated using technology provided by the processor IP owners. There is a companion document (OVP Guide to Using Processor Models) which explains the general concepts of OVP Fast Processor Models and their use. It is downloadable from the OVPworld website documentation pages.

1.1 Description

RISC-V D45 64-bit processor model

1.2 Licensing

This Model is released under the Open Source Apache 2.0

1.3 Extensions

1.3.1 Extensions Enabled by Default

The model has the following architectural extensions enabled, and the corresponding bits in the `misa` CSR Extensions field will be set upon reset:

- `misa` bit 0: extension A (atomic instructions)
- `misa` bit 2: extension C (compressed instructions)
- `misa` bit 3: extension D (double-precision floating point)
- `misa` bit 5: extension F (single-precision floating point)
- `misa` bit 8: RV32I/RV64I/RV128I base integer instruction set
- `misa` bit 12: extension M (integer multiply/divide instructions)
- `misa` bit 13: extension N (user-level interrupts)
- `misa` bit 18: extension S (Supervisor mode)
- `misa` bit 20: extension U (User mode)
- `misa` bit 23: extension X (non-standard extensions present)

To specify features that can be dynamically enabled or disabled by writes to the `misa` register in addition to those listed above, use parameter “`add_Extensions_mask`”. This is a string parameter containing the feature letters to add; for example, value “DV” indicates that double-precision floating point and the Vector Extension can be enabled or disabled by writes to the `misa` register, if supported on this variant. Parameter “`sub_Extensions_mask`” can be used to disable dynamic update of features in the same way.

Legacy parameter “`misa_Extensions_mask`” can also be used. This `Uns32`-valued parameter specifies all writable bits in the `misa` Extensions field, replacing any permitted bits defined in the base variant.

Note that any features that are indicated as present in the `misa` mask but absent in the `misa` will be ignored. See the next section.

1.3.2 Enabling Other Extensions

The following extensions are supported by the model, but not enabled by default in this variant:

- `misa` bit 1: extension B (bit manipulation extension)
- `misa` bit 4: RV32E base integer instruction set (embedded)
- `misa` bit 7: extension H (hypervisor)
- `misa` bit 10: extension K (cryptographic)
- `misa` bit 15: extension P (DSP instructions)
- `misa` bit 21: extension V (vector extension)

To add features from this list to the visible set in the `misa` register, use parameter “`add_Extensions`”. This is a string containing identification letters of features to enable; for example, value “`DV`” indicates that double-precision floating point and the Vector Extension should be enabled, if they are currently absent and are available on this variant.

Legacy parameter “`misa_Extensions`” can also be used. This `Uns32`-valued parameter specifies the reset value for the `misa` CSR `Extensions` field, replacing any permitted bits defined in the base variant.

To add features from this list to the implicitly-enabled set (not visible in the `misa` register), use parameter “`add_implicit_Extensions`”. This is a string parameter in the same format as the “`add_Extensions`” parameter described above.

1.3.3 Disabling Extensions

The following extensions are enabled by default in the model and can be disabled:

`misa` bit 0: extension A (atomic instructions)

`misa` bit 2: extension C (compressed instructions)

`misa` bit 3: extension D (double-precision floating point)

`misa` bit 5: extension F (single-precision floating point)

`misa` bit 12: extension M (integer multiply/divide instructions)

`misa` bit 13: extension N (user-level interrupts)

`misa` bit 18: extension S (Supervisor mode)

`misa` bit 20: extension U (User mode)

`misa` bit 23: extension X (non-standard extensions present)

To disable features that are enabled by default, use parameter “`sub_Extensions`”. This is a string containing identification letters of features to disable; for example, value “`DF`” indicates that double-precision and single-precision floating point extensions should be disabled, if they are enabled by default on this variant.

To remove features from this list from the implicitly-enabled set (not visible in the `misa` register), use parameter “`sub_implicit_Extensions`”. This is a string parameter in the same format as the “`sub_Extensions`” parameter described above.

1.4 General Features

1.4.1 `mtvec` CSR

On this variant, the Machine trap-vector base-address register (`mtvec`) is writable. It can instead be configured as read-only using parameter “`mtvec_is_ro`”.

Values written to “`mtvec`” are masked using the value `0xffffffffffffc`. A different mask of writable

bits may be specified using parameter “`mtvec_mask`” if required. In addition, when Vectored interrupt mode is enabled, parameter “`tvec_align`” may be used to specify additional hardware-enforced base address alignment. In this variant, “`tvec_align`” defaults to 0, implying no alignment constraint.

If parameter “`mtvec_sext`” is True, values written to “`mtvec`” are sign-extended from the most-significant writable bit. In this variant, “`mtvec_sext`” is False, indicating that “`mtvec`” is not sign-extended.

The initial value of “`mtvec`” is 0x0. A different value may be specified using parameter “`mtvec`” if required.

1.4.2 `stvec` CSR

Values written to “`stvec`” are masked using the value 0xffffffffffffc. A different mask of writable bits may be specified using parameter “`stvec_mask`” if required. In addition, when Vectored interrupt mode is enabled, parameter “`tvec_align`” may be used to specify additional hardware-enforced base address alignment. In this variant, “`tvec_align`” defaults to 0, implying no alignment constraint.

If parameter “`stvec_sext`” is True, values written to “`stvec`” are sign-extended from the most-significant writable bit. In this variant, “`stvec_sext`” is False, indicating that “`stvec`” is not sign-extended.

1.4.3 `utvec` CSR

Values written to “`utvec`” are masked using the value 0xffffffffffffc. A different mask of writable bits may be specified using parameter “`utvec_mask`” if required. In addition, when Vectored interrupt mode is enabled, parameter “`tvec_align`” may be used to specify additional hardware-enforced base address alignment. In this variant, “`tvec_align`” defaults to 0, implying no alignment constraint.

If parameter “`utvec_sext`” is True, values written to “`utvec`” are sign-extended from the most-significant writable bit. In this variant, “`utvec_sext`” is False, indicating that “`utvec`” is not sign-extended.

1.4.4 Reset

On reset, the model will restart at address 0x0. A different reset address may be specified using parameter “`reset_address`” or applied using optional input port “`reset_addr`” if required.

1.4.5 NMI

On an NMI, the model will restart at address 0x0; a different NMI address may be specified using parameter “`nmi_address`” or applied using optional input port “`nmi_addr`” if required. The cause reported on an NMI is 0x0 by default; a different cause may be specified using parameter “`ecode_nmi`” or applied using optional input port “`nmi_cause`” if required.

If parameter “rnmi_version” is not “none”, resumable NMIs are supported, managed by additional CSRs “mnscratch”, “mnepc”, “mncause” and “mnstatus”, following the indicated version of the Resumable NMI extension proposal. In this variant, “rnmi_version” is “none”.

The NMI input is level-sensitive. To instead specify that the NMI input is latched on the rising edge of the NMI signal, set parameter “nmi_is_latched” to True.

1.4.6 WFI

WFI will halt the processor until an interrupt occurs. It can instead be configured as a NOP using parameter “wfi_is_nop”. WFI timeout wait is implemented with a time limit of 0 (i.e. WFI causes an Illegal Instruction trap in Supervisor mode when mstatus.TW=1).

1.4.7 cycle CSR

The “cycle” CSR is implemented in this variant. Set parameter “cycle_undefined” to True to instead specify that “cycle” is unimplemented and accesses should cause Illegal Instruction traps.

1.4.8 instret CSR

The “instret” CSR is implemented in this variant. Set parameter “instret_undefined” to True to instead specify that “instret” is unimplemented and accesses should cause Illegal Instruction traps.

1.4.9 hpmcounter CSR

The “hpmcounter” CSRs are implemented in this variant. Set parameter “hpmcounter_undefined” to True to instead specify that “hpmcounter” CSRs are unimplemented and accesses should cause Illegal Instruction traps.

1.4.10 time CSR

The “time” CSR is implemented in this variant. Set parameter “time_undefined” to True to instead specify that “time” is unimplemented and reads of it should cause Illegal Instruction traps. Usually, the value of the “time” CSR should be provided by the platform - see notes below about the artifact “CSR” bus for information about how this is done.

1.4.11 mcycle CSR

The “mcycle” CSR is implemented in this variant. Set parameter “mcycle_undefined” to True to instead specify that “mcycle” is unimplemented and accesses should cause Illegal Instruction traps.

1.4.12 minstret CSR

The “minstret” CSR is implemented in this variant. Set parameter “minstret_undefined” to True to instead specify that “minstret” is unimplemented and accesses should cause Illegal Instruction traps.

1.4.13 mhpmcounter CSR

The “mhpmcounter” CSRs are implemented in this variant. Set parameter “mhpmcounter_undefined” to True to instead specify that “mhpmcounter” CSRs are unimplemented and accesses should cause Illegal Instruction traps.

1.4.14 Virtual Memory

This variant supports address translation modes 0 (bare), 8 (Sv39), 9 (Sv48) and 10 (Sv57). Use parameter “Sv_modes” to specify a bit mask of different implemented modes if required; for example, setting “Sv_modes” to $(1 << 0) + (1 << 8)$ indicates that mode 0 (bare) and mode 8 (Sv39) are implemented. These indices correspond to writable values in the satp.MODE CSR field.

A 0-bit ASID is implemented. Use parameter “ASID_bits” to specify a different implemented ASID size if required.

TLB behavior is controlled by parameter “ASIDCacheSize”. If this parameter is 0, then an unlimited number of TLB entries will be maintained concurrently. If this parameter is non-zero, then only TLB entries for up to “ASIDCacheSize” different ASIDs will be maintained concurrently initially; as new ASIDs are used, TLB entries for less-recently used ASIDs are deleted, which improves model performance in some cases. If the model detects that the TLB entry cache is too small (entry ejections are very frequent), it will increase the cache size automatically. In this variant, “ASIDCacheSize” is 8.

1.4.15 Unaligned Accesses

Unaligned memory accesses are supported by this variant. Set parameter “unaligned” to “F” to disable such accesses.

Unaligned memory accesses are not supported for AMO instructions by this variant. Set parameter “unalignedAMO” to “T” to enable such accesses.

Address misaligned exceptions are higher priority than page fault or access fault exceptions on this variant. Set parameter “unaligned_low_pri” to “T” to specify that they are lower priority instead.

1.4.16 PMP

A PMP unit is not implemented by this variant. Set parameter “PMP_registers” to indicate that the unit should be implemented with that number of PMP entries.

Accesses to unimplemented PMP registers are write-ignored and read as zero on this variant. Set

parameter “PMP_undefined” to True to indicate that such accesses should cause Illegal Instruction exceptions instead.

1.4.17 LR/SC Granule

LR/SC instructions are implemented with a 1-byte reservation granule. A different granule size may be specified using parameter “lr_sc_grain”.

1.5 Compressed Extension

Standard compressed instructions are present in this variant. Legacy compressed extension features may also be configured using parameters described below. Use parameter “commcompress_version” to enable more recent compressed extension features if required.

Parameter Zcea_version is used to specify the version of Zcea instructions present. By default, Zcea_version is set to “none” in this variant. Updates to this parameter require a commercial product license.

Parameter Zceb_version is used to specify the version of Zceb instructions present. By default, Zceb_version is set to “none” in this variant. Updates to this parameter require a commercial product license.

Parameter Zcee_version is used to specify the version of Zcee instructions present. By default, Zcee_version is set to “none” in this variant. Updates to this parameter require a commercial product license.

1.6 Floating Point Features

The D extension is enabled in this variant independently of the F extension. Set parameter “d_requires_f” to “T” to specify that the D extension requires the F extension to be enabled.

Half precision floating point is not implemented. Use parameter “Zfh” to enable this if required.

By default, the processor starts with floating-point instructions disabled (mstatus.FS=0). Use parameter “mstatus_FS” to force mstatus.FS to a non-zero value for floating-point to be enabled from the start.

The specification is imprecise regarding the conditions under which mstatus.FS is set to Dirty state (3). Parameter “mstatus_fs_mode” can be used to specify the required behavior in this model, as described below.

If “mstatus_fs_mode” is set to “always_dirty” then the model implements a simplified floating point status view in which mstatus.FS holds values 0 (Off) and 3 (Dirty) only; any write of values 1 (Initial) or 2 (Clean) from privileged code behave as if value 3 was written.

If “mstatus_fs_mode” is set to “write.1” then mstatus.FS will be set to 3 (Dirty) by any explicit write to the fflags, frm or fcsr control registers, or by any executed instruction that writes an FPR, or by any executed floating point compare or conversion to integer/unsigned that signals a floating

point exception. Floating point compare or conversion to integer/unsigned instructions that do not signal an exception will not set `mstatus.FS`.

If “`mstatus.fs_mode`” is set to “`write_any`” then `mstatus.FS` will be set to 3 (Dirty) by any explicit write to the `fflags`, `frm` or `fcsr` control registers, or by any executed instruction that writes an FPR, or by any executed floating point compare or conversion even if those instructions do not signal a floating point exception.

In this variant, “`mstatus.fs_mode`” is set to “`write_1`”.

1.7 Privileged Architecture

This variant implements the Privileged Architecture with version specified in the References section of this document. Note that parameter “`priv_version`” can be used to select the required architecture version; see the following sections for detailed information about differences between each supported version.

1.7.1 Legacy Version 1.10

1.10 version of May 7 2017.

1.7.2 Version 20190608

Stable 1.11 version of June 8 2019, with these changes compared to version 1.10:

- `mcountinhibit` CSR defined;
- pages are never executable in Supervisor mode if page table entry U bit is 1;
- `mstatus.TW` is writable if any lower-level privilege mode is implemented (previously, it was just if Supervisor mode was implemented);

1.7.3 Version 20211203

1.12 draft version of December 3 2021, with these changes compared to version 20190608:

- `mstatush`, `mseccfg`, `mseccfgh`, `menvcfg`, `menvcfgh`, `senvcfg`, `henvcfg`, `henvcfgh` and `mconfigptr` CSRs defined;
- `xret` instructions clear `mstatus.MPRV` when leaving Machine mode if new mode is less privileged than M-mode;
- maximum number of PMP registers increased to 64;
- data endian is now configurable.

1.7.4 Version 1.12

Official 1.12 version, identical to 20211203.

1.7.5 Version master

Unstable master version, currently identical to 1.12.

1.8 Unprivileged Architecture

This variant implements the Unprivileged Architecture with version specified in the References section of this document. Note that parameter “user_version” can be used to select the required architecture version; see the following sections for detailed information about differences between each supported version.

1.8.1 Legacy Version 2.2

2.2 version of May 7 2017.

1.8.2 Version 20191213

Stable 20191213-Base-Ratified version of December 13 2019, with these changes compared to version 2.2:

- floating point fmin/fmax instruction behavior modified to comply with IEEE 754-201x.
- numerous other optional behaviors can be separately enabled using Z-prefixed parameters.

1.9 Other Extensions

Other extensions that can be configured are described in this section.

1.9.1 Zmmul

Parameter “Zmmul” is 0 on this variant, meaning that all multiply and divide instructions are implemented. if “Zmmul” is set to 1 then multiply instructions are implemented but divide and remainder instructions are not implemented.

1.9.2 Zicsr

Parameter “Zicsr” is 1 on this variant, meaning that standard CSRs and CSR access instructions are implemented. if “Zicsr” is set to 0 then standard CSRs and CSR access instructions are not implemented and an alternative scheme must be provided as a processor extension.

1.9.3 Zifencei

Parameter “Zifencei” is 1 on this variant, meaning that the fence.i instruction is implemented (but treated as a NOP by the model). if “Zifencei” is set to 0 then the fence.i instruction is not implemented.

1.9.4 Zicbom

Parameter “Zicbom” is 0 on this variant, meaning that code block management instructions are undefined. if “Zicbom” is set to 1 then code block management instructions cbo.clean, cbo.flush and cbo.inval are defined.

If Zicbom is present, the cache block size is given by parameter “cmomp_bytes”. The instructions may cause traps if used illegally but otherwise are NOPs in this model.

1.9.5 Zicbop

Parameter “Zicbop” is 0 on this variant, meaning that prefetch instructions are undefined. if “Zicbop” is set to 1 then prefetch instructions prefetch.i, prefetch.r and prefetch.w are defined (but behave as NOPs in this model).

1.9.6 Zicboz

Parameter “Zicboz” is 0 on this variant, meaning that the cbo.zero instruction is undefined. if “Zicboz” is set to 1 then the cbo.zero instruction is defined.

If Zicboz is present, the cache block size is given by parameter “cmoz_bytes”.

1.9.7 Svnapot

Parameter “Svnapot_page_mask” is 0x0 on this variant, meaning that NAPOT Translation Contiguity is not implemented. if “Svnapot_page_mask” is non-zero then NAPOT Translation Contiguity is enabled for page sizes indicated by that mask value when page table entry bit 63 is set.

If Svnapot is present, “Svnapot_page_mask” is a mask of page sizes for which contiguous pages can be created. For example, a value of 0x10000 implies that 64KiB contiguous pages are supported.

1.9.8 Svpbmt

Parameter “Svpbmt” is 0 on this variant, meaning that page-based memory types are not implemented. if “Svpbmt” is set to 1 then page-based memory types are indicated by page table entry bits 62:61.

Note that except for their effect on Page Faults, the encoded memory types do not alter the behavior of this model, which always implements strongly-ordered non-cacheable semantics.

1.9.9 Svinval

Parameter “Svinval” is 0 on this variant, meaning that fine-grained address-translation cache invalidation instructions are not implemented. If “Svinval” is set to 1 then fine-grained address-translation cache invalidation instructions `sinval.vma`, `sfence.w.inval` and `sfence.inval.ir` are implemented.

1.9.10 Smstateen

Parameter “Smstateen” is 0 on this variant, meaning that state enable CSRs are undefined. If “Smstateen” is set to 1 then state enable CSRs are defined.

Within the state enable CSRs, only bit 1 (for `Zfinx`), bit 57 (for `xcontext` CSR access), bit 62 (for `xenvcfg` CSR access) and bit 63 (for lower-level state enable CSR access) are currently implemented.

1.10 CLIC

The model can be configured to implement a Core Local Interrupt Controller (CLIC) using parameter “CLICLEVELS”; when non-zero, the CLIC is present with the specified number of interrupt levels (2-256), as described in the RISC-V Core-Local Interrupt Controller specification, and further parameters are made available to configure other aspects of the CLIC. “CLICLEVELS” is zero in this variant, indicating that a CLIC is not implemented.

1.11 Load-Reserved/Store-Conditional Locking

By default, LR/SC locking is implemented automatically by the model and simulator, with a reservation granule defined by the “`lr_sc_grain`” parameter. It is also possible to implement locking externally to the model in a platform component, using the “`LR_address`”, “`SC_address`” and “`SC_valid`” net ports, as described below.

The “`LR_address`” output net port is written by the model with the address used by a load-reserved instruction as it executes. This port should be connected as an input to the external lock management component, which should record the address, and also that an LR/SC transaction is active.

The “`SC_address`” output net port is written by the model with the address used by a store-conditional instruction as it executes. This should be connected as an input to the external lock management component, which should compare the address with the previously-recorded load-reserved address, and determine from this (and other implementation-specific constraints) whether the store should succeed. It should then immediately write the Boolean success/fail code to the “`SC_valid`” input net port of the model. Finally, it should update state to indicate that an LR/SC transaction is no longer active.

It is also possible to write zero to the “`SC_valid`” input net port at any time outside the context of a store-conditional instruction, which will mark any active LR/SC transaction as invalid.

Irrespective of whether LR/SC locking is implemented internally or externally, taking any exception or interrupt or executing exception-return instructions (e.g. MRET) will always mark any active LR/SC transaction as invalid.

Parameter “amo_aborts_lr_sc” is used to specify whether AMO operations abort any active LR/SC pair. In this variant, “amo_aborts_lr_sc” is 0.

1.12 Active Atomic Operation Indication

The “AMO_active” output net port is written by the model with a code indicating any current atomic memory operation while the instruction is active. The written codes are:

0: no atomic instruction active

1: AMOMIN active

2: AMOMAX active

3: AMOMINU active

4: AMOMAXU active

5: AMOADD active

6: AMOXOR active

7: AMOOR active

8: AMOAND active

9: AMOSWAP active

10: LR active

11: SC active

1.13 Interrupts

The “reset” port is an active-high reset input. The processor is halted when “reset” goes high and resumes execution from the reset address specified using the “reset_address” parameter or “reset_addr” port when the signal goes low. The “mcause” register is cleared to zero.

The “nmi” port is an active-high NMI input. The processor resumes execution from the address specified using the “nmi_address” parameter or “nmi_addr” port when the NMI signal goes high. The “mcause” register is cleared to zero.

All other interrupt ports are active high. For each implemented privileged execution level, there are by default input ports for software interrupt, timer interrupt and external interrupt; for example, for Machine mode, these are called “MSWInterrupt”, “MTimerInterrupt” and “MExternalInterrupt”, respectively. When the N extension is implemented, ports are also present for User mode. Parameter “unimp_int_mask” allows the default behavior to be changed to exclude certain interrupt ports. The parameter value is a mask in the same format as the “mip” CSR; any interrupt

corresponding to a non-zero bit in this mask will be removed from the processor and read as zero in “mip”, “mie” and “mideleg” CSRs (and Supervisor and User mode equivalents if implemented).

Parameter “external_int_id” can be used to enable extra interrupt ID input ports on each hart. If the parameter is True then when an external interrupt is applied the value on the ID port is sampled and used to fill the Exception Code field in the “mcause” CSR (or the equivalent CSR for other execution levels). For Machine mode, the extra interrupt ID port is called “MExternalInterruptID”.

The “deferint” port is an active-high artifact input that, when written to 1, prevents any pending-and-enabled interrupt being taken (normally, such an interrupt would be taken on the next instruction after it becomes pending-and-enabled). The purpose of this signal is to enable alignment with hardware models in step-and-compare usage.

1.14 Debug Mode

The model can be configured to implement Debug mode using parameter “debug_mode”. This implements features described in Chapter 4 of the RISC-V External Debug Support specification with version specified by parameter “debug_version” (see References). Some aspects of this mode are not defined in the specification because they are implementation-specific; the model provides infrastructure to allow implementation of a Debug Module using a custom harness. Features added are described below.

Parameter “debug_mode” can be used to specify three different behaviors, as follows:

1. If set to value “vector”, then operations that would cause entry to Debug mode result in the processor jumping to the address specified by the “debug_address” parameter. It will execute at this address, in Debug mode, until a “dret” instruction causes return to non-Debug mode. Any exception generated during this execution will cause a jump to the address specified by the “dexc_address” parameter.
2. If set to value “interrupt”, then operations that would cause entry to Debug mode result in the processor simulation call (e.g. `opProcessorSimulate`) returning, with a stop reason of `OP_SR_INTERRUPT`. In this usage scenario, the Debug Module is implemented in the simulation harness.
3. If set to value “halt”, then operations that would cause entry to Debug mode result in the processor halting. Depending on the simulation environment, this might cause a return from the simulation call with a stop reason of `OP_SR_HALT`, or debug mode might be implemented by another platform component which then restarts the debugged processor again.

1.14.1 Debug State Entry

The specification does not define how Debug mode is implemented. In this model, Debug mode is enabled by a Boolean pseudo-register, “DM”. When “DM” is True, the processor is in Debug mode. When “DM” is False, mode is defined by “mstatus” in the usual way.

Entry to Debug mode can be performed in any of these ways:

1. By writing True to register “DM” (e.g. using `opProcessorRegWrite`) followed by simulation of

at least one cycle (e.g. using `opProcessorSimulate`), `dcsr` cause will be reported as trigger;

2. By writing a 1 then 0 to net “`haltreq`” (using `opNetWrite`) followed by simulation of at least one cycle (e.g. using `opProcessorSimulate`);
3. By writing a 1 to net “`resethaltreq`” (using `opNetWrite`) while the “`reset`” signal undergoes a negege transition, followed by simulation of at least one cycle (e.g. using `opProcessorSimulate`);
4. By executing an “`ebreak`” instruction when Debug mode entry for the current processor mode is enabled by `dcsr.ebreakm`, `dcsr.ebreaks` or `dcsr.ebreaku`.

In all cases, the processor will save required state in “`dpc`” and “`dcsr`” and then perform actions described above, depending in the value of the “`debug_mode`” parameter.

1.14.2 Debug State Exit

Exit from Debug mode can be performed in any of these ways:

1. By writing `False` to register “`DM`” (e.g. using `opProcessorRegWrite`) followed by simulation of at least one cycle (e.g. using `opProcessorSimulate`);
2. By executing an “`dret`” instruction when Debug mode.

In both cases, the processor will perform the steps described in section 4.6 (Resume) of the Debug specification.

1.14.3 Debug Registers

When Debug mode is enabled, registers “`dcsr`”, “`dpc`”, “`dscratch0`” and “`dscratch1`” are implemented as described in the specification. These may be manipulated externally by a Debug Module using `opProcessorRegRead` or `opProcessorRegWrite`; for example, the Debug Module could write “`dcsr`” to enable “`ebreak`” instruction behavior as described above, or read and write “`dpc`” to emulate stepping over an “`ebreak`” instruction prior to resumption from Debug mode.

1.14.4 Debug Mode Execution

The specification allows execution of code fragments in Debug mode. A Debug Module implementation can cause execution in Debug mode by the following steps:

1. Write the address of a Program Buffer to the program counter using `opProcessorPCSet`;
2. If “`debug_mode`” is set to “`halt`”, write 0 to pseudo-register “`DMStall`” (to leave halted state);
3. If entry to Debug mode was handled by exiting the simulation callback, call `opProcessorSimulate` or `opRootModuleSimulate` to resume simulation.

Debug mode will be re-entered in these cases:

1. By execution of an “`ebreak`” instruction; or:
2. By execution of an instruction that causes an exception.

In both cases, the processor will either jump to the debug exception address, or return control immediately to the harness, with `stopReason` of `OP_SR_INTERRUPT`, or perform a halt, depending on the value of the “`debug_mode`” parameter.

1.14.5 Debug Single Step

When in Debug mode, the processor or harness can cause a single instruction to be executed on return from that mode by setting `dcsr.step`. After one non-Debug-mode instruction has been executed, control will be returned to the harness. The processor will remain in single-step mode until `dcsr.step` is cleared.

1.14.6 Debug Event Priorities

The model supports two different models for determining which debug exception occurs when multiple debug events are pending:

- 1: original mode (when parameter “`debug_priority`”=“`original`”);
- 2: modified mode, as described in Debug Specification pull request 693 (when parameter “`debug_priority`”=“`PR693`”). This mode resolves some anomalous behavior of the original specification.

1.14.7 Debug Ports

Port “`DM`” is an output signal that indicates whether the processor is in Debug mode

Port “`haltreq`” is a rising-edge-triggered signal that triggers entry to Debug mode (see above).

Port “`resethaltreq`” is a level-sensitive signal that triggers entry to Debug mode after reset (see above).

1.15 Trigger Module

This model is configured with a trigger module, implementing a subset of the behavior described in Chapter 5 of the RISC-V External Debug Support specification with version specified by parameter “`debug_version`” (see References).

1.15.1 Trigger Module Restrictions

The model currently supports `tdata1` of type 0, type 2 (`mcontrol`), type 3 (`icount`), type 4 (`itrigger`), type 5 (`etrigger`) and type 6 (`mcontrol6`). `icount` triggers are implemented for a single instruction only, with count hard-wired to 1 and automatic zeroing of mode bits when the trigger fires.

1.15.2 Trigger Module Parameters

Parameter “trigger_num” is used to specify the number of implemented triggers. In this variant, “trigger_num” is 4.

Parameter “tinfo” is used to specify the value of the read-only “tinfo” register, which indicates the trigger types supported. In this variant, “tinfo” is 0x3d.

Parameter “tinfo_undefined” is used to specify whether the “tinfo” register is undefined, in which case reads of it trap to Machine mode. In this variant, “tinfo_undefined” is 0.

Parameter “tcontrol_undefined” is used to specify whether the “tcontrol” register is undefined, in which case accesses to it trap to Machine mode. In this variant, “tcontrol_undefined” is 0.

Parameter “mcontext_undefined” is used to specify whether the “mcontext” register is undefined, in which case accesses to it trap to Machine mode. In this variant, “mcontext_undefined” is 0.

Parameter “scontext_undefined” is used to specify whether the “scontext” register is undefined, in which case accesses to it trap to Machine mode. In this variant, “scontext_undefined” is 0.

Parameter “amo_trigger” is used to specify whether load/store triggers are activated for AMO instructions. In this variant, “amo_trigger” is 0.

Parameter “no_hit” is used to specify whether the “hit” bit in tdata1 is unimplemented. In this variant, “no_hit” is 0.

Parameter “no_sselect_2” is used to specify whether the “sselect” field in “textra32”/“textra64” registers is unable to hold value 2 (indicating match by ASID is not allowed). In this variant, “no_sselect_2” is 0.

Parameter “mcontext_bits” is used to specify the number of writable bits in the “mcontext” register. In this variant, “mcontext_bits” is 13.

Parameter “scontext_bits” is used to specify the number of writable bits in the “scontext” register. In this variant, “scontext_bits” is 34.

Parameter “mvalue_bits” is used to specify the number of writable bits in the “mvalue” field in “textra32”/“textra64” registers; if zero, the “mselect” field is tied to zero. In this variant, “mvalue_bits” is 13.

Parameter “svalue_bits” is used to specify the number of writable bits in the “svalue” field in “textra32”/“textra64” registers; if zero, the “sselect” is tied to zero. In this variant, “svalue_bits” is 34.

Parameter “mcontrol_maskmax” is used to specify the value of field “maskmax” in the “mcontrol” register. In this variant, “mcontrol_maskmax” is 63.

1.16 Debug Mask

It is possible to enable model debug messages in various categories. This can be done statically using the “debugflags” parameter, or dynamically using the “debugflags” command. Enabled messages are specified using a bitmask value, as follows:

Value 0x002: enable debugging of PMP and virtual memory state;

Value 0x004: enable debugging of interrupt state.

All other bits in the debug bitmask are reserved and must not be set to non-zero values.

1.17 Integration Support

This model implements a number of non-architectural pseudo-registers and other features to facilitate integration.

1.17.1 CSR Register External Implementation

If parameter “enable_CSR_bus” is True, an artifact 16-bit bus “CSR” is enabled. Slave callbacks installed on this bus can be used to implement modified CSR behavior (use opBusSlaveNew or icmMapExternalMemory, depending on the client API). A CSR with index 0xABC is mapped on the bus at address 0xABC0; as a concrete example, implementing CSR “time” (number 0xC01) externally requires installation of callbacks at address 0xC010 on the CSR bus.

1.17.2 LR/SC Active Address

Artifact register “LRSCAddress” shows the active LR/SC lock address. The register holds all-ones if there is no LR/SC operation active or if LR/SC locking is implemented externally as described above.

1.17.3 Page Table Walk Introspection

Artifact register “PTWStage” shows the active page table translation stage (0 if no stage active, 1 if HS-stage active, 2 if VS-stage active and 3 if G-stage active). This register is visibly non-zero only in a memory access callback triggered by a page table walk event.

Artifact register “PTWInputAddr” shows the input address of active page table translation. This register is visibly non-zero only in a memory access callback triggered by a page table walk event.

Artifact register “PTWLevel” shows the active level of page table translation (corresponding to index variable “i” in the algorithm described by Virtual Address Translation Process in the RISC-V Privileged Architecture specification). This register is visibly non-zero only in a memory access callback triggered by a page table walk event.

1.17.4 Artifact Register “fflags_i”

If parameter “enable_fflags_i” is True, an 8-bit artifact register “fflags_i” is added to the model. This register shows the floating point flags set by the current instruction (unlike the standard “fflags” CSR, in which the flag bits are sticky).

1.18 Limitations

Instruction pipelines are not modeled in any way. All instructions are assumed to complete immediately. This means that instruction barrier instructions (e.g. fence.i) are treated as NOPs, with the exception of any Illegal Instruction behavior, which is modeled.

Caches and write buffers are not modeled in any way. All loads, fetches and stores complete immediately and in order, and are fully synchronous. Data barrier instructions (e.g. fence) are treated as NOPs, with the exception of any Illegal Instruction behavior, which is modeled.

Real-world timing effects are not modeled: all instructions are assumed to complete in a single cycle.

Hardware Performance Monitor registers are not implemented and hardwired to zero.

The TLB is architecturally-accurate but not device accurate. This means that all TLB maintenance and address translation operations are fully implemented but the cache is larger than in the real device.

This variant is under development. It may not be complete.

Andes-specific cache, local memory and ECC behavior is not yet implemented, except for CSR state.

Andes Performance and Code Dense instructions and associated CSR state are implemented, but the EXEC.IT instruction supports in-memory table mode using the uitb CSR only (not hardwired mode).

PMP and PMA accesses that any-byte match but do not all-byte match are broken into separate smaller accesses that follow all-byte match rules.

1.19 Verification

All instructions have been extensively tested by Imperas, using tests generated specifically for this model and also reference tests from <https://github.com/riscv/riscv-tests>.

Also reference tests have been used from various sources including:

<https://github.com/riscv/riscv-tests>

<https://github.com/ucb-bar/riscv-torture>

The Imperas OVPsim RISC-V models are used in the RISC-V Foundation Compliance Framework as a functional Golden Reference:

<https://github.com/riscv/riscv-compliance>

where the simulated model is used to provide the reference signatures for compliance testing. The Imperas OVPsim RISC-V models are used as reference in both open source and commercial instruction stream test generators for hardware design verification, for example:

<http://valtrix.in/sting> from Valtrix

<https://github.com/google/riscv-dv> from Google

The Imperas OVPsim RISC-V models are also used by commercial and open source RISC-V Core RTL developers as a reference to ensure correct functionality of their IP.

1.20 References

The Model details are based upon the following specifications:

RISC-V Instruction Set Manual, Volume I: User-Level ISA (User Architecture Version 2.2)

RISC-V Instruction Set Manual, Volume II: Privileged Architecture (Privileged Architecture Version Ratified-IMFDQC-and-Priv-v1.11)

RISC-V External Debug Support (RISC-V External Debug Support Version 0.13.2-DRAFT)

— This is an initial configuration for the variant

Chapter 2

Andes-Specific Extensions

Andes processors add various custom extensions to the basic RISC-V architecture. This model implements the following:

- 1: Hardware Stack Protection (if `mmisc_cfg.HSP=1`);
- 2: Physical Memory Attribute Unit (if `mmisc_cfg.DPMA=1`).
- 3: Performance Throttling (register interface only, if `mmisc_cfg.PFT=1`);
- 4: CSRs for CCTL Operations (register interface only, if `mmisc_cfg.CCTLCSR=1`);
- 5: Performance Extension instructions (if `mmisc_cfg.EV5MPE=1`);
- 6: CodeDense instructions (if `mmisc_cfg.ECD=1`);
- 7: Half-precision load/store instructions (if `mmisc_cfg.EFHW=1`).
- 8: BFLOAT16 conversion instructions (if `mmisc_cfg.BFLOAT16=1`).
- 9: Half-precision arithmetic instructions (if `mmisc_cfg.ZFH=1`).
- 10: Vector INT4 load extension (if `mmisc_cfg.VL4=1`).
- 11: Vector packed FP16 extension (if `mmisc_cfg.VPFH=1`).

Other Andes-specific extensions are not currently modeled. The exact set of supported extensions can be configured using parameter “`andesExtensions/mmisc_cfg`”, which overrides the default value of the `mmisc_cfg` register (see detailed description below).

2.1 Andes-Specific Parameters

In addition to the base model RISC-V parameters, this model implements parameters allowing Andes-specific model features to be controlled. These parameters are documented below.

2.1.1 Parameter `andesExtensions/mmsc_cfg`

This parameter allows the value of the read-only `mmsc_cfg` register to be specified. Bits that affect behavior of the model are:

bit 3 (ECD): enables CodeDense instructions and `uitb` CSR.

bit 4 (PFT): determines presence of `mpft_ctl` register and affects implemented fields in `mxstatus`.

bit 5 (HSP): enables HW Stack protection, relevant CSRs and affects implemented fields in `mxstatus`.

bit 12 (VPLIC): enables Vectored Interrupts support.

bit 13 (EV5PE): enables Performance Extension support.

bit 14 (LMSLVP): enables Local Memory slave ports `ILM_port` and `DLM_port`.

bit 15 (PMNDS): enables Andes-enhanced Performance Monitoring.

bit 16 (CCTLCSR): enables CCTL CSRs.

bit 30 (DPMA): enables the Physical Memory Attribute Unit and relevant CSRs.

bit 32 (BF16CVT): enables BFLOAT16 conversion extension.

bit 33 (ZFH): enables FP16 half-precision extension.

bit 34 (VL4): enables vector INT4 load extension.

bit 44 (VPFH): enables vector packed FP16 extension.

bit 45 (L2CMP_CFG): enables cluster configuration fields. `CORE_PCLUS` field will be set to `floor(numharts-1, 1)`.

bit 46 (L2C): enables `ml2c_ctl_base` CSR if both L2C and L2CMP_CFG are not zero

Other bits can be set or cleared but do not affect model behavior.

Example: `-override iss/cpu0/andesExtensions/mmsc_cfg=0x2028`

2.1.2 Parameter `andesExtensions/micm_cfg`

This parameter allows the value of the read-only `micm_cfg` register to be specified. Bits that affect behavior of the model are:

bits 8:6 (ISZ): enables `mcache_ctl` CSR if non-zero.

bits 14:12 (ILMB): enables `milmb` CSR if non-zero.

bits 19:15 (ILMSZ): specifies size of ILM in KB if non-zero (ILM size = $1024 \ll (\text{ILMSZ}-1)$)

Other bits can be set or cleared but do not affect model behavior, except that if any bit is non zero then IME/PIME bits in `mxstatus` are modeled.

Example: `-override iss/cpu0/andesExtensions/micm_cfg=0`

2.1.3 Parameter `andesExtensions/mdcm_cfg`

This parameter allows the value of the read-only `mdcm_cfg` register to be specified. Bits that affect behavior of the model are:

bits 8:6 (DSZ): enables `mcache_ctl` CSR if non-zero.

bits 14:12 (DLMB): enables `mdlmb` CSR if non-zero.

bits 19:15 (DLMSZ): specifies size of ILM in KB if non-zero (DLM size = $1024 \ll (\text{DLMSZ}-1)$)

Other bits can be set or cleared but do not affect model behavior, except that if any bit is non zero then DME/DIME bits in `mxstatus` are modeled.

Example: `-override iss/cpu0/andesExtensions/mdcm_cfg=0`

2.1.4 Parameter `andesExtensions/uitb`

This parameter allows the value of the `uitb` register to be specified.

Example: `-override iss/cpu0/andesExtensions/uitb=0`

2.1.5 Parameter `andesExtensions/milmb`

This parameter allows the value of the `milmb` register to be specified.

Example: `-override iss/cpu0/andesExtensions/milmb=0x200001`

2.1.6 Parameter `andesExtensions/milmbMask`

This parameter allows the mask of writable bits in the `milmb` register to be specified. The default value for this variant is `0xe` (RWECC and ECCEN are writable, all other bits are read-only).

Example: `-override iss/cpu0/andesExtensions/milmbMask=0xe`

2.1.7 Parameter `andesExtensions/mdlmb`

This parameter allows the value of the `mdlmb` register to be specified.

Example: `-override iss/cpu0/andesExtensions/mdlmb==0x300001`

2.1.8 Parameter `andesExtensions/mdlmbMask`

This parameter allows the mask of writable bits in the `mdlmb` register to be specified. The default value for this variant is `0xe` (RWECC and ECCEN are writable, all other bits are read-only).

Example: `-override iss/cpu0/andesExtensions/mdlmbMask=0xe`

2.1.9 Parameter `andesExtensions/PMA_grain`

This parameter allows the grain size of Physical Memory Attribute regions to be specified. The default value for this variant is 0, meaning that PMA regions as small as 4 bytes are implemented.

Example: `-override iss/cpu0/andesExtensions/PMA_grain=16`

2.2 Hardware Stack Protection

Hardware Stack Protection is present on this variant (`mmsc_cfg.HSP=1`). Registers `mhsp_ctl`, `mmp_bound` and `mmp_base` are implemented.

2.3 Physical Memory Attribute Unit

The Physical Memory Attribute Unit is not present on this variant (`mmsc_cfg.DPMA=0`).

2.4 Performance Throttling

Performance Throttling registers are present on this variant (`mmsc_cfg.PFT=1`). Register `mpft_ctl` is present but has no behavior except for the effects on `mxstatus`, which are modeled.

2.5 Andes-Enhanced Performance Monitoring

Andes-Enhanced Performance Monitoring is present on this variant (`mmsc_cfg.PMNDS=1`).

2.6 CSRs for CCTL Operations

CSRs for CCTL Operation are present on this variant (`mmsc_cfg.CCTLCSR=1`) but have no effect except that trap behavior for illegal use is modeled.

2.7 Andes-Specific Instructions

This section describes Andes-specific instructions implemented by this variant. Refer to Andes reference documentation for more information.

2.7.1 Performance Extension Instructions

2.7.1.1 ADDIGP

| | | | | | | | | | |
|---------|----|-----------|----|---------|----|-----------------|----|------------|--|
| 31 | 30 | 21 | 20 | 19 | 17 | 16 | 15 | | |
| imm[17] | | imm[10:1] | | imm[11] | | imm[14:12] | | imm[16:15] | |
| 14 | 13 | 12 | 11 | 7 | 6 | 0 | | | |
| imm[0] | | 01 | | Rd | | Custom0 0001011 | | | |

Add the content of the implied GP (x3) register with a signed constant.

2.7.1.2 BBC

| | | | | | | | | | |
|---------|----|----------|----|----------|----|-----------------|----|-----|--|
| 31 | 30 | 29 | 25 | 24 | 20 | 19 | 15 | | |
| imm[10] | | 0 | | imm[9:5] | | cimm[4:0] | | Rs1 | |
| 14 | 12 | 11 | 8 | 7 | 6 | 0 | | | |
| 111 | | imm[4:1] | | cimm[5] | | Custom2 1011011 | | | |

Branch on bit is clear/zero.

2.7.1.3 BBS

| | | | | | | | | | |
|---------|----|----------|----|----------|----|-----------------|----|-----|--|
| 31 | 30 | 29 | 25 | 24 | 20 | 19 | 15 | | |
| imm[10] | | 1 | | imm[9:5] | | cimm[4:0] | | Rs1 | |
| 14 | 12 | 11 | 8 | 7 | 6 | 0 | | | |
| 111 | | imm[4:1] | | cimm[5] | | Custom2 1011011 | | | |

Branch on bit is set/non-zero.

2.7.1.4 BEQC

| | | | | | | | | | |
|---------|----|----------|----|----------|----|-----------------|----|-----|--|
| 31 | 30 | 29 | 25 | 24 | 20 | 19 | 15 | | |
| imm[10] | | cimm[6] | | imm[9:5] | | cimm[4:0] | | Rs1 | |
| 14 | 12 | 11 | 8 | 7 | 6 | 0 | | | |
| 101 | | imm[4:1] | | cimm[5] | | Custom2 1011011 | | | |

Branch on equal to a constant.

2.7.1.5 BNEC

| | | | | | | | | | |
|---------|----|----------|----|----------|----|-----------------|----|-----|--|
| 31 | 30 | 29 | 25 | 24 | 20 | 19 | 15 | | |
| imm[10] | | cimm[6] | | imm[9:5] | | cimm[4:0] | | Rs1 | |
| 14 | 12 | 11 | 8 | 7 | 6 | 0 | | | |
| 110 | | imm[4:1] | | cimm[5] | | Custom2 1011011 | | | |

Branch on not-equal to a constant.

2.7.1.6 BFOS

| | | | | | | | | | | | |
|----------|----|----------|----|-----|----|-----|----|----|---|--------------------|---|
| 31 | 26 | 25 | 20 | 19 | 15 | 14 | 12 | 11 | 7 | 6 | 0 |
| msb[5:0] | | lsb[5:0] | | Rs1 | | 011 | | Rd | | Custom2 1011011 | |

Sign-extended bit-field extract or insert operation.

2.7.1.7 BFOZ

| | | | | | | | | | | | |
|----------|----|----------|----|-----|----|-----|----|----|---|--------------------|---|
| 31 | 26 | 25 | 20 | 19 | 15 | 14 | 12 | 11 | 7 | 6 | 0 |
| msb[5:0] | | lsb[5:0] | | Rs1 | | 010 | | Rd | | Custom2 1011011 | |

Zero-extended bit-field extract or insert operation.

2.7.1.8 LEA.h

| | | | | | | | | | | | |
|---------|----|-----|----|-----|----|-----|----|----|---|--------------------|---|
| 31 | 25 | 24 | 20 | 19 | 15 | 14 | 12 | 11 | 7 | 6 | 0 |
| 0000101 | | Rs2 | | Rs1 | | 000 | | Rd | | Custom2 1011011 | |

Add a base register with a half-word-aligned offset from an offset register.

2.7.1.9 LEA.w

| | | | | | | | | | | | |
|---------|----|-----|----|-----|----|-----|----|----|---|--------------------|---|
| 31 | 25 | 24 | 20 | 19 | 15 | 14 | 12 | 11 | 7 | 6 | 0 |
| 0000110 | | Rs2 | | Rs1 | | 000 | | Rd | | Custom2 1011011 | |

Add a base register with a word-aligned offset from an offset register.

2.7.1.10 LEA.d

| | | | | | | | | | | | |
|---------|----|-----|----|-----|----|-----|----|----|---|--------------------|---|
| 31 | 25 | 24 | 20 | 19 | 15 | 14 | 12 | 11 | 7 | 6 | 0 |
| 0000111 | | Rs2 | | Rs1 | | 000 | | Rd | | Custom2 1011011 | |

Add a base register with a double-word-aligned offset from an offset register.

2.7.1.11 LEA.b.ze

| | | | | | | | | | | | |
|---------|----|-----|----|-----|----|-----|----|----|---|--------------------|---|
| 31 | 25 | 24 | 20 | 19 | 15 | 14 | 12 | 11 | 7 | 6 | 0 |
| 0001000 | | Rs2 | | Rs1 | | 000 | | Rd | | Custom2 1011011 | |

Add a base register with an unsigned 32-bit byte offset from an offset register.

2.7.1.12 LEA.h.ze

| | | | | | | | | | | | |
|---------|----|-----|----|-----|----|-----|----|----|---|--------------------|---|
| 31 | 25 | 24 | 20 | 19 | 15 | 14 | 12 | 11 | 7 | 6 | 0 |
| 0001001 | | Rs2 | | Rs1 | | 000 | | Rd | | Custom2 1011011 | |

Add a base register with an unsigned 32-bit half-word offset from an offset register.

2.7.1.13 LEA.w.ze

| | | | | | | | | | | | |
|---------|----|-----|----|-----|----|-----|----|----|---|--------------------|---|
| 31 | 25 | 24 | 20 | 19 | 15 | 14 | 12 | 11 | 7 | 6 | 0 |
| 0001010 | | Rs2 | | Rs1 | | 000 | | Rd | | Custom2 1011011 | |

Add a base register with an unsigned 32-bit word offset from an offset register.

2.7.1.14 LEA.d.ze

| | | | | | | | | | | | |
|---------|----|-----|----|-----|----|-----|----|----|---|--------------------|---|
| 31 | 25 | 24 | 20 | 19 | 15 | 14 | 12 | 11 | 7 | 6 | 0 |
| 0001011 | | Rs2 | | Rs1 | | 000 | | Rd | | Custom2 1011011 | |

Add a base register with an unsigned 32-bit double-word offset from an offset register.

2.7.1.15 LBGP

| | | | | | | | | | |
|---------|----|-----------|----|---------|----|-----------------|----|------------|--|
| 31 | 30 | 21 | 20 | 19 | 17 | 16 | 15 | | |
| imm[17] | | imm[10:1] | | imm[11] | | imm[14:12] | | imm[16:15] | |
| 14 | 13 | 12 | 11 | 7 | 6 | 0 | | | |
| imm[0] | | 00 | | Rd | | Custom0 0001011 | | | |

Load a sign-extended 8-bit byte from memory into a general register.

2.7.1.16 LBUGP

| | | | | | | | | | |
|---------|----|-----------|----|---------|----|-----------------|----|------------|--|
| 31 | 30 | 21 | 20 | 19 | 17 | 16 | 15 | | |
| imm[17] | | imm[10:1] | | imm[11] | | imm[14:12] | | imm[16:15] | |
| 14 | 13 | 12 | 11 | 7 | 6 | 0 | | | |
| imm[0] | | 10 | | Rd | | Custom0 0001011 | | | |

Load a zero-extended 8-bit byte from memory into a general register.

2.7.1.17 LHGP

| | | | | | | | |
|------------|----|-----------|----|---------|----|-----------------|---|
| 31 | 30 | 21 | 20 | 19 | 17 | | |
| imm[17] | | imm[10:1] | | imm[11] | | imm[14:12] | |
| 16 | 15 | 14 | 12 | 11 | 7 | 6 | 0 |
| imm[16:15] | | 001 | | Rd | | Custom1 0101011 | |

Load a sign-extended 16-bit half-word from memory into a general register.

2.7.1.18 LHUGP

| | | | | | |
|------------|-----------|-----|---------|------------|-----------------|
| 31 | 30 | 21 | 20 | 19 | 17 |
| imm[17] | imm[10:1] | | imm[11] | imm[14:12] | |
| 16 | 15 | 14 | 12 | 11 | 7 |
| imm[16:15] | | 101 | | Rd | Custom1 0101011 |

Load a zero-extended 16-bit half-word from memory into a general register.

2.7.1.19 LWGP

| | | | | | | |
|------------|-----------|-----|---------|---------|-----------------|----|
| 31 | 30 | 22 | 21 | 20 | 19 | 17 |
| imm[18] | imm[10:2] | | imm[17] | imm[11] | imm[14:12] | |
| 16 | 15 | 14 | 12 | 11 | 7 | 6 |
| imm[16:15] | | 010 | | Rd | Custom1 0101011 | |

Load a sign-extended 32-bit word from memory into a general register.

2.7.1.20 LWUGP

| | | | | | | |
|------------|-----------|-----|---------|---------|-----------------|----|
| 31 | 30 | 22 | 21 | 20 | 19 | 17 |
| imm[18] | imm[10:2] | | imm[17] | imm[11] | imm[14:12] | |
| 16 | 15 | 14 | 12 | 11 | 7 | 6 |
| imm[16:15] | | 110 | | Rd | Custom1 0101011 | |

Load a zero-extended 32-bit word from memory into a general register.

2.7.1.21 LDGP

| | | | | | | | |
|------------|-----------|-----|------------|----|-----------------|------------|----|
| 31 | 30 | 23 | 22 | 21 | 20 | 19 | 17 |
| imm[19] | imm[10:3] | | imm[18:17] | | imm[11] | imm[14:12] | |
| 16 | 15 | 14 | 12 | 11 | 7 | 6 | 0 |
| imm[16:15] | | 011 | | Rd | Custom1 0101011 | | |

Load a 64-bit double-word from memory into a general register.

2.7.1.22 SBGP

| | | | | | | | | |
|---------|-----------|----|----|----------|------------|--------------------|------------|----|
| 31 | 30 | 25 | 24 | 20 | 19 | 17 | 16 | 15 |
| imm[17] | imm[10:5] | | | Rs2 | imm[14:12] | | imm[16:15] | |
| 14 | 13 | 12 | 11 | 8 | 7 | 6 | 0 | 0 |
| imm[0] | | 11 | | imm[4:1] | imm[11] | Custom0 0001011 | | |

Store an 8-bit byte from a general register into a memory location.

2.7.1.23 SHGP

| | | | | | | | | | |
|---------|----|-----------|----|---------|----|-----------------|----|------------|--|
| 31 | 30 | 25 | 24 | 20 | 19 | 17 | 16 | 15 | |
| imm[17] | | imm[10:5] | | Rs2 | | imm[14:12] | | imm[16:15] | |
| 14 | 12 | 11 | 8 | 7 | 6 | 0 | | | |
| 000 | | imm[4:1] | | imm[11] | | Custom1 0101011 | | | |

Store a 16-bit half-word from a general register into a memory location.

2.7.1.24 SWGP

| | | | | | | | | | |
|---------|----|-----------|----|---------|----|------------|----|-----------------|--|
| 31 | 30 | 25 | 24 | 20 | 19 | 17 | 16 | 15 | |
| imm[18] | | imm[10:5] | | Rs2 | | imm[14:12] | | imm[16:15] | |
| 14 | 12 | 11 | 9 | 8 | 7 | 6 | 0 | | |
| 100 | | imm[4:2] | | imm[17] | | imm[11] | | Custom1 0101011 | |

Store a 32-bit word from a general register into a memory location.

2.7.1.25 SDGP

| | | | | | | | | | |
|---------|----|-----------|----|------------|----|------------|----|-----------------|--|
| 31 | 30 | 25 | 24 | 20 | 19 | 17 | 16 | 15 | |
| imm[19] | | imm[10:5] | | Rs2 | | imm[14:12] | | imm[16:15] | |
| 14 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 0 | |
| 111 | | imm[4:3] | | imm[18:17] | | imm[11] | | Custom1 0101011 | |

Store a 64-bit double-word from a general register into a memory location.

2.7.1.26 FFB

| | | | | | | | | | | | |
|---------|----|-----|----|-----|----|-----|----|----|---|-----------------|---|
| 31 | 25 | 24 | 20 | 19 | 15 | 14 | 12 | 11 | 7 | 6 | 0 |
| 0010000 | | Rs2 | | Rs1 | | 000 | | Rd | | Custom2 1011011 | |

Find the first byte in a first register that matches a value in a second register.

2.7.1.27 FFZMISM

| | | | | | | | | | | | |
|---------|----|-----|----|-----|----|-----|----|----|---|-----------------|---|
| 31 | 25 | 24 | 20 | 19 | 15 | 14 | 12 | 11 | 7 | 6 | 0 |
| 0010001 | | Rs2 | | Rs1 | | 000 | | Rd | | Custom2 1011011 | |

Find the first byte in a register that is zero or fails a corresponding byte comparison.

2.7.1.28 FFMISM

| | | | | | | | | | | | |
|---------|----|-----|----|-----|----|-----|----|----|---|--------------------|---|
| 31 | 25 | 24 | 20 | 19 | 15 | 14 | 12 | 11 | 7 | 6 | 0 |
| 0010010 | | Rs2 | | Rs1 | | 000 | | Rd | | Custom2 1011011 | |

Find the first byte in a register that fails a corresponding byte comparison.

2.7.1.29 FLMISM

| | | | | | | | | | | | |
|---------|----|-----|----|-----|----|-----|----|----|---|--------------------|---|
| 31 | 25 | 24 | 20 | 19 | 15 | 14 | 12 | 11 | 7 | 6 | 0 |
| 0010011 | | Rs2 | | Rs1 | | 000 | | Rd | | Custom2 1011011 | |

Find the last byte in a register that fails a corresponding byte comparison.

2.7.2 CodeDense Instructions

2.7.2.1 EXEC.IT

| | | | | | | | | | | | |
|-----|----|---------------|---|---------|---|---|---|----------------|---|----|--|
| 15 | 13 | 12 | 9 | 8 | 7 | 6 | 2 | 1 | 0 | | |
| 100 | | imm[10 4:3 8] | | imm[11] | | 0 | | imm[7:6 2 9 5] | | 00 | |

Execute an instruction fetched from the instruction table.

2.7.2.2 EX9.IT

| | | | | | | | | | |
|-----|----|---------------|---|----|---|----------------|---|----|---|
| 15 | 13 | 12 | 9 | 8 | 7 | 6 | 2 | 1 | 0 |
| 100 | | imm[10 4:3 8] | | 00 | | imm[7:6 2 9 5] | | 00 | |

Execute an instruction fetched from the instruction table.

2.7.3 Half-Precision Load/Store Instructions

2.7.3.1 FLHW

| | | | | | | | | | |
|-----------|----|-----|----|-----|----|-----|---|--------------------|---|
| 31 | 20 | 19 | 15 | 14 | 12 | 11 | 7 | 6 | 0 |
| imm[11:0] | | Rs1 | | 000 | | FRd | | LOAD-FP 0000111 | |

Load half-precision floating-point value from memory and convert it to a single-precision value.

2.7.3.2 FSHW

| | | | | | | | | | | | |
|-----------|----|------|----|-----|----|-----|----|----------|---|---------------------|---|
| 31 | 25 | 24 | 20 | 19 | 15 | 14 | 12 | 11 | 7 | 6 | 0 |
| imm[11:5] | | FRs2 | | Rs1 | | 000 | | imm[4:0] | | STORE-FP 0100111 | |

Store half-precision floating-point data to memory. The half-precision floating-point value is converted from a single-precision value source.

2.8 Andes Net Port Names

Net ports in this model use generic base model names and not Andes-specific names. Equivalent port names for Andes nets are listed below:

“reset_n”: equivalent to “reset”

“reset_vector”: equivalent to “reset_addr”

“meip”: equivalent to “MExternalInterrupt”

“meiid”: equivalent to “MExternalInterruptID”

“meiack”: equivalent to “MExternalInterruptACK”

“mtip”: equivalent to “MTimerInterrupt”

“stip”: equivalent to “MSWInterrupt”

“nmi”: equivalent to “nmi”

“seip”: equivalent to “SExternalInterrupt”

“seiid”: equivalent to “SExternalInterruptID”

“seiack”: equivalent to “SExternalInterruptACK”

“stip”: equivalent to “STimerInterrupt”

“stip”: equivalent to “SSWInterrupt”

“ueip”: equivalent to “UExternalInterrupt”

“ueiid”: equivalent to “UExternalInterruptID”

“ueiack”: equivalent to “UExternalInterruptACK”

Chapter 3

Configuration

3.1 Location

This model's VLVN is `andes.ovpworld.org/processor/riscv/1.0`.

The model source is usually at:

`$IMPERAS_HOME/ImperasLib/source/andes.ovpworld.org/processor/riscv/1.0`

The model binary is usually at:

`$IMPERAS_HOME/lib/$IMPERAS_ARCH/ImperasLib/andes.ovpworld.org/processor/riscv/1.0`

3.2 GDB Path

The default GDB for this model is: `$IMPERAS_HOME/lib/$IMPERAS_ARCH/gdb/riscv-none-embed-gdb`.

3.3 Semi-Host Library

The default semi-host library file is `riscv.ovpworld.org/semihosting/pk/1.0`

3.4 Processor Endian-ness

This is a LITTLE endian model.

3.5 QuantumLeap Support

This processor is qualified to run in a QuantumLeap enabled simulator.

3.6 Processor ELF code

The ELF code supported by this model is: `0xf3`.

Chapter 4

All Variants in this model

This model has these variants

| Variant | Description |
|-----------------------|------------------------------|
| N25 | |
| NX25 | |
| N25F | |
| NX25F | |
| A25 | |
| AX25 | |
| A25F | |
| AX25F | |
| NX27V | |
| N22 | |
| A27 | |
| A45 | |
| AX27 | |
| AX45 | |
| AX45MP _x 1 | |
| AX45MP _x 2 | |
| AX45MP _x 4 | |
| D25F | |
| D45 | (described in this document) |
| NX45 | |
| N45 | |
| N25F-SE | |

Table 4.1: All Variants in this model

Chapter 5

Bus Master Ports

This model has these bus master ports.

| Name | min | max | Connect? | Description |
|-------------|-----|-----|-----------|-----------------|
| INSTRUCTION | 32 | 64 | mandatory | Instruction bus |
| DATA | 32 | 64 | optional | Data bus |

Table 5.1: Bus Master Ports

Chapter 6

Bus Slave Ports

This model has no bus slave ports.

Chapter 7

Net Ports

This model has these net ports.

| Name | Type | Connect? | Description |
|----------------------|--------|----------|--|
| reset | input | optional | Reset |
| reset_addr | input | optional | externally-applied reset address |
| nmi | input | optional | NMI |
| nmi_cause | input | optional | externally-applied NMI cause |
| nmi_addr | input | optional | externally-applied NMI address |
| USWInterrupt | input | optional | User software interrupt |
| SSWInterrupt | input | optional | Supervisor software interrupt |
| MSWInterrupt | input | optional | Machine software interrupt |
| UTimerInterrupt | input | optional | User timer interrupt |
| STimerInterrupt | input | optional | Supervisor timer interrupt |
| MTimerInterrupt | input | optional | Machine timer interrupt |
| UExternalInterrupt | input | optional | User external interrupt |
| UExternalInterruptID | input | optional | User external interrupt ID (sampled if non-zero as interrupt taken to User mode) |
| SExternalInterrupt | input | optional | Supervisor external interrupt |
| SExternalInterruptID | input | optional | Supervisor external interrupt ID (sampled if non-zero as interrupt taken to Supervisor mode) |
| MExternalInterrupt | input | optional | Machine external interrupt |
| MExternalInterruptID | input | optional | Machine external interrupt ID (sampled if non-zero as interrupt taken to Machine mode) |
| irq_ack_o | output | optional | interrupt acknowledge (pulse) |
| irq_id_o | output | optional | acknowledged interrupt id (valid during irq_ack_o pulse) |
| sec_lvl_o | output | optional | current privilege level |
| LR_address | output | optional | Port written with effective address for LR instruction |
| SC_address | output | optional | Port written with effective address for SC instruction |
| SC_valid | input | optional | SC_address valid input signal |

| | | | |
|-----------------------|--------|----------|---|
| AMO_active | output | optional | Port written with code indicating active AMO |
| deferint | input | optional | Artifact signal causing interrupts to be held off when high |
| MExternalInterruptACK | output | optional | Machine mode external interrupt acknowledge |
| SExternalInterruptACK | output | optional | Supervisor mode external interrupt acknowledge |
| UExternalInterruptACK | output | optional | User mode external interrupt acknowledge |

Table 7.1: Net Ports

Chapter 8

FIFO Ports

This model has no FIFO ports.

Chapter 9

Formal Parameters

| Name | Type | Description |
|------------------------------|-------------|---|
| Fundamental | | |
| variant | Enumeration | Selects variant (either a generic UISA or a specific model) |
| user_version | Enumeration | Specify required User Architecture version (2.2, 2.3, 20190305 or 20191213) |
| priv_version | Enumeration | Specify required Privileged Architecture version (1.10, 1.11, 20190405, 20190608, 20211203, 1.12 or master) |
| numHarts | Uns32 | Specify the number of hart contexts in a multiprocessor |
| endian | Endian | Model endian |
| enable_expanded | Boolean | Specify that 48-bit and 64-bit expanded instructions are supported |
| endianFixed | Boolean | Specify that data endianness is fixed (mstatus.{MBE,SBE,UBE} fields are read-only) |
| misa_MXL | Uns32 | Override default value of misa.MXL |
| misa_Extensions | Uns32 | Override default value of misa.Extensions |
| add_Extensions | String | Add extensions specified by letters to misa.Extensions (for example, specify “VD” to add V and D features) |
| sub_Extensions | String | Remove extensions specified by letters from misa.Extensions (for example, specify “VD” to remove V and D features) |
| misa_Extensions_mask | Uns32 | Override mask of writable bits in misa.Extensions |
| add_Extensions_mask | String | Add extensions specified by letters to mask of writable bits in misa.Extensions (for example, specify “VD” to add V and D features) |
| sub_Extensions_mask | String | Remove extensions specified by letters from mask of writable bits in misa.Extensions (for example, specify “VD” to remove V and D features) |
| add_implicit_Extensions | String | Add extensions specified by letters to implicitly-present extensions not visible in misa.Extensions |
| sub_implicit_Extensions | String | Remove extensions specified by letters from implicitly-present extensions not visible in misa.Extensions |
| Compressed_Extension | | |
| compress_version | Enumeration | Specify required Compressed Architecture version (legacy, 0.70.1 or 0.70.5) |
| Zcea_version | Enumeration | Specify version of Zcea implemented (legacy only) (none or 0.50.1) |
| Zceb_version | Enumeration | Specify version of Zceb implemented (legacy only) (none or 0.50.1) |
| Zcee_version | Enumeration | Specify version of Zcee implemented (legacy only) (none or 1.0.0-rc) |
| Debug_Extension | | |
| debug_version | Enumeration | Specify required Debug Architecture version (0.13.2-DRAFT, 0.14.0-DRAFT or 1.0.0-STABLE) |
| debug_mode | Enumeration | Specify how Debug mode is implemented (none, vector, interrupt or halt) |
| lr_sc_constraint | Enumeration | Specify memory constraint for LR/SC instructions (none, user1 or user2) |
| amo_constraint | Enumeration | Specify memory constraint for AMO instructions (none, user1 or user2) |
| Interrupts_Exceptions | | |
| rnmi_version | Enumeration | Specify required RNMI Architecture version (none or 0.2.1) |

| | | |
|----------------------------|-------------|--|
| mtvec_is_ro | Boolean | Specify whether mtvec CSR is read-only |
| tvec_align | Uns32 | Specify hardware-enforced alignment of mtvec/stvec/utvec when Vectored interrupt mode enabled |
| ecode_mask | Uns64 | Specify hardware-enforced mask of writable bits in xcause.ExceptionCode |
| ecode_nmi | Uns64 | Specify xcause.ExceptionCode for NMI |
| nmi_is_latched | Boolean | Specify whether NMI input is latched on rising edge (if False, it is level-sensitive) |
| tval_zero | Boolean | Specify whether mtval/stval/utval are hard wired to zero |
| tval_zero_ebreak | Boolean | Specify whether mtval/stval/utval are set to zero by an ebreak |
| tval_ii_code | Boolean | Specify whether mtval/stval contain faulting instruction bits on illegal instruction exception |
| trap_preserves_lr | Boolean | Whether a trap preserves active LR/SC state |
| xret_preserves_lr | Boolean | Whether an xret instruction preserves active LR/SC state |
| reset_address | Uns64 | Override reset vector address |
| nmi_address | Uns64 | Override NMI vector address |
| CLINT_address | Uns64 | Specify base address of internal CLINT model (or 0 for no CLINT) |
| local_int_num | Uns32 | Specify number of supplemental local interrupts |
| unimp_int_mask | Uns64 | Specify mask of unimplemented interrupts (e.g. 1<<9 indicates Supervisor external interrupt unimplemented) |
| force_mideleg | Uns64 | Specify mask of interrupts always delegated to lower-priority execution level from Machine execution level |
| force_sideleg | Uns64 | Specify mask of interrupts always delegated to User execution level from Supervisor execution level |
| no_ideleg | Uns64 | Specify mask of interrupts that cannot be delegated to lower-priority execution levels |
| no_e deleg | Uns64 | Specify mask of exceptions that cannot be delegated to lower-priority execution levels |
| external_int_id | Boolean | Whether to add nets allowing External Interrupt ID codes to be forced |
| Floating Point | | |
| mstatus_fs_mode | Enumeration | Specify conditions causing update of mstatus.FS to dirty (write_1, write_any, always_dirty or force_dirty) |
| d_requires_f | Boolean | If D and F extensions are separately enabled in the misa CSR, whether D is enabled only if F is enabled |
| enable_fflags_i | Boolean | Whether fflags.i artifact register present (shows per-instruction floating point flags) |
| mstatus_FS | Uns32 | Override default value of mstatus.FS (initial state of floating point unit) |
| Zfh | Boolean | Specify that Zfh is implemented (IEEE half-precision floating point is supported) |
| Zfhmin | Boolean | Specify that Zfhmin is implemented (restricted IEEE half-precision floating point is supported) |
| Zfinx_version | Enumeration | Specify version of Zfinx implemented (use integer register file for floating point instructions) (none, 0.4 or 0.41) |
| Simulation Artifact | | |
| use_hw_reg_names | Boolean | Specify whether to use hardware register names x0-x31 and f0-f31 instead of ABI register names |
| no_pseudo_inst | Boolean | Specify whether pseudo-instructions should not be reported in trace and disassembly |
| ABI_d | Boolean | Specify whether D registers are used for parameters (ABI SemiHosting) |
| verbose | Boolean | Specify verbose output messages |
| traceVolatile | Boolean | Specify whether volatile registers (e.g. minstret) should be shown in change trace |
| enable_CSR_bus | Boolean | Add artifact CSR bus port, allowing CSR registers to be externally implemented |
| CSR_remap | String | Comma-separated list of CSR number mappings, each of the form <csr-Name>=<number> |

| | | |
|---------------------------------|---------|---|
| ASID_cache_size | Uns32 | Specifies the number of different ASIDs for which TLB entries are cached; a value of 0 implies no limit |
| Memory | | |
| updatePTEA | Boolean | Specify whether hardware update of PTE A bit is supported |
| updatePTED | Boolean | Specify whether hardware update of PTE D bit is supported |
| unaligned_low_pri | Boolean | Specify whether address misaligned exceptions are lower priority than page or access fault exceptions |
| unaligned | Boolean | Specify whether the processor supports unaligned memory accesses |
| unalignedAMO | Boolean | Specify whether the processor supports unaligned memory accesses for AMO instructions |
| amo_aborts_lr_sc | Boolean | Specify whether AMO operations abort any active LR/SC pair |
| ASID_bits | Uns32 | Specify the number of implemented ASID bits |
| lr_sc_grain | Uns32 | Specify byte granularity of ll/sc lock region (constrained to a power of two) |
| Sv_modes | Uns32 | Specify bit mask of implemented address translation modes (e.g. (1<<0)+(1<<8) indicates “bare” and “Sv39” modes may be selected in satp.MODE) |
| Instruction_CSR_Behavior | | |
| wfi_is_nop | Boolean | Specify whether WFI should be treated as a NOP (if not, halt while waiting for interrupts) |
| counteren_mask | Uns32 | Specify hardware-enforced mask of writable bits in mcounteren/scounteren registers |
| noinhibit_mask | Uns32 | Specify hardware-enforced mask of always-zero bits in mcountinhibit register |
| cycle_undefined | Boolean | Specify that the cycle CSR is undefined |
| mcycle_undefined | Boolean | Specify that the mcycle CSR is undefined |
| time_undefined | Boolean | Specify that the time CSR is undefined |
| instret_undefined | Boolean | Specify that the instret CSR is undefined |
| minstret_undefined | Boolean | Specify that the minstret CSR is undefined |
| hpmcounter_undefined | Boolean | Specify that the hpmcounter CSRs are undefined |
| mhpmcounter_undefined | Boolean | Specify that the mhpmcounter CSRs are undefined |
| CSR Masks | | |
| mtvec_mask | Uns64 | Specify hardware-enforced mask of writable bits in mtvec register |
| stvec_mask | Uns64 | Specify hardware-enforced mask of writable bits in stvec register |
| utvec_mask | Uns64 | Specify hardware-enforced mask of writable bits in utvec register |
| tdata1_mask | Uns64 | Specify hardware-enforced mask of writable bits in Trigger Module tdata1 register |
| mip_mask | Uns64 | Specify hardware-enforced mask of writable bits in mip register |
| sip_mask | Uns64 | Specify hardware-enforced mask of writable bits in sip register |
| uip_mask | Uns64 | Specify hardware-enforced mask of writable bits in uip register |
| mtvec_sext | Boolean | Specify whether mtvec is sign-extended from most-significant bit |
| stvec_sext | Boolean | Specify whether stvec is sign-extended from most-significant bit |
| utvec_sext | Boolean | Specify whether utvec is sign-extended from most-significant bit |
| MXL_writable | Boolean | Specify that misa.MXL is writable (feature under development) |
| SXL_writable | Boolean | Specify that mstatus.SXL is writable (feature under development) |
| UXL_writable | Boolean | Specify that mstatus.UXL is writable (feature under development) |
| Trigger | | |
| tinfo_undefined | Boolean | Specify that the tinfo CSR is undefined |
| tcontrol_undefined | Boolean | Specify that the tcontrol CSR is undefined |
| mcontext_undefined | Boolean | Specify that the mcontext CSR is undefined |
| scontext_undefined | Boolean | Specify that the scontext CSR is undefined |
| mscontext_undefined | Boolean | Specify that the mscontext CSR is undefined (Debug Version 0.14.0 and later) |
| amo_trigger | Boolean | Specify whether AMO load/store operations activate triggers |
| no_hit | Boolean | Specify that tdata1.hit is unimplemented |
| no_sselect_2 | Boolean | Specify that textra.sselect=2 is not supported (no trigger match by ASID) |

| | | |
|--------------------------|---------|---|
| trigger_num | Uns32 | Specify the number of implemented hardware triggers |
| tinfo | Uns32 | Override tinfo register (for all triggers) |
| mcontext_bits | Uns32 | Specify the number of implemented bits in mcontext |
| scontext_bits | Uns32 | Specify the number of implemented bits in scontext |
| mvalue_bits | Uns32 | Specify the number of implemented bits in textra.mvalue (if zero, textra.mselect is tied to zero) |
| svalue_bits | Uns32 | Specify the number of implemented bits in textra.svalue (if zero, textra.sselect is tied to zero) |
| mcontrol_maskmax | Uns32 | Specify mcontrol.maskmax value |
| PMP Configuration | | |
| PMP_grain | Uns32 | Specify PMP region granularity, G (0 =>4 bytes, 1 =>8 bytes, etc) |
| PMP_registers | Uns32 | Specify the number of implemented PMP address registers |
| PMP_max_page | Uns32 | Specify the maximum size of PMP region to map if non-zero (may improve performance; constrained to a power of two) |
| PMP_decompose | Boolean | Whether unaligned PMP accesses are decomposed into separate aligned accesses |
| PMP_undefined | Boolean | Whether accesses to unimplemented PMP registers are undefined (if True) or write ignored and zero (if False) |
| PMP_maskparams | Boolean | Enable parameters to change the read-only masks for PMP CSRs |
| PMP_initialparams | Boolean | Enable parameters to change the reset values for PMP CSRs |
| Other Extensions | | |
| Svnapot_page_mask | Uns64 | Specify mask of implemented Svnapot intermediate page sizes (e.g. 1<<16 means 64KiB contiguous regions are supported) |
| Smstateen | Boolean | Specify that Smstateen is implemented |
| Svpbmt | Boolean | Specify that Svpbmt is implemented |
| Svinal | Boolean | Specify that Svinal is implemented |
| Zicr | Boolean | Specify that Zicr is implemented |
| Zifencei | Boolean | Specify that Zifencei is implemented |
| Zicbom | Boolean | Specify that Zicbom is implemented |
| Zicbop | Boolean | Specify that Zicbop is implemented |
| Zicboz | Boolean | Specify that Zicboz is implemented |
| Zmmul | Boolean | Specify that Zmmul is implemented |
| CSR Defaults | | |
| mvendorid | Uns64 | Override mvendorid register |
| marchid | Uns64 | Override marchid register |
| mimpid | Uns64 | Override mimpid register |
| mhartid | Uns64 | Override mhartid register (or first mhartid of an incrementing sequence if this is an SMP variant) |
| mtvec | Uns64 | Override mtvec register |
| Fast Interrupt | | |
| CLICLEVELS | Uns32 | Specify number of interrupt levels implemented by CLIC, or 0 if CLIC absent |

Table 9.1: Parameters that can be set in: Hart

9.1 Extension Parameters

| Name | Type | Description |
|-----------|-------|---|
| PMA_grain | Uns32 | Specify PMA region granularity, G (0 =>4 bytes, 1 =>8 bytes, etc) |
| milmb | Uns64 | Override milmb register |
| mdlmb | Uns64 | Override mdlmb register |
| mmsc_cfg | Uns64 | Override mmsc_cfg register |
| micm_cfg | Uns64 | Override micm_cfg register |
| mdcm_cfg | Uns64 | Override mdcm_cfg register |
| mvec_cfg | Uns32 | Override mvec_cfg register (ignored if mdcm_cfg.veccfg=0) |

| | | |
|---------------|--------|---|
| uitb | Uns64 | Override uitb register |
| ml2c_ctl_base | Uns64 | Override ml2c_ctl_base register (ignored if mmisc_cfg.L2CMP_CFG and .L2C are 0) |
| milmbMask | Uns64 | Override milmb register writable bit mask |
| mdlmbMask | Uns64 | Override mdlmb register writable bit mask |
| aceFile | String | Specify ACE extension shared object |
| aceLibrary | String | Specify ACE COPILOT simulation shared object |
| aceDiagLevel | Uns32 | Specify ACE Interface function diagnostic level (0->3) |

Table 9.2: Parameters for andesExtensions

9.2 Parameters with enumerated types

9.2.1 Parameter user_version

| Set to this value | Description |
|-------------------|---------------------------------------|
| 2.2 | User Architecture Version 2.2 |
| 2.3 | Deprecated and equivalent to 20191213 |
| 20190305 | Deprecated and equivalent to 20191213 |
| 20191213 | User Architecture Version 20191213 |

Table 9.3: Values for Parameter user_version

9.2.2 Parameter priv_version

| Set to this value | Description |
|-------------------|--|
| 1.10 | Privileged Architecture Version 1.10 |
| 1.11 | Privileged Architecture Version 1.11, equivalent to 20190608 |
| 20190405 | Deprecated and equivalent to 20190608 |
| 20190608 | Privileged Architecture Version Ratified-IMFDQC-and-Priv-v1.11 |
| 20211203 | Privileged Architecture Version 20211203 |
| 1.12 | Privileged Architecture Version 1.12, equivalent to 20211203 |
| master | Privileged Architecture Master Branch as of commit 6bdeb58 (this is subject to change) |

Table 9.4: Values for Parameter priv_version

9.2.3 Parameter compress_version

| Set to this value | Description |
|-------------------|--|
| legacy | Compressed Architecture absent or legacy version |
| 0.70.1 | Compressed Architecture Version 0.70.1 |
| 0.70.5 | Compressed Architecture Version 0.70.5 |

Table 9.5: Values for Parameter compress_version

9.2.4 Parameter debug_version

| Set to this value | Description |
|-------------------|--|
| 0.13.2-DRAFT | RISC-V External Debug Support Version 0.13.2-DRAFT |
| 0.14.0-DRAFT | RISC-V External Debug Support Version 0.14.0-DRAFT |
| 1.0.0-STABLE | RISC-V External Debug Support Version 1.0.0-STABLE |

Table 9.6: Values for Parameter debug_version

9.2.5 Parameter `rnmi_version`

| Set to this value | Description |
|-------------------|----------------------|
| none | RNMI not implemented |
| 0.2.1 | RNMI version 0.2.1 |

Table 9.7: Values for Parameter `rnmi_version`

9.2.6 Parameter `mstatus_fs_mode`

| Set to this value | Description |
|---------------------------|---|
| <code>write_1</code> | Any non-zero flag result sets <code>mstatus.fs</code> dirty |
| <code>write_any</code> | Any write of flags sets <code>mstatus.fs</code> dirty |
| <code>always_dirty</code> | <code>mstatus.fs</code> is either off or dirty |
| <code>force_dirty</code> | <code>mstatus.fs</code> is forced to dirty |

Table 9.8: Values for Parameter `mstatus_fs_mode`

9.2.7 Parameter `debug_mode`

| Set to this value | Description |
|-------------------|---|
| none | Debug mode not implemented |
| vector | Debug mode implemented by execution at vector |
| interrupt | Debug mode implemented by interrupt |
| halt | Debug mode implemented by halt |

Table 9.9: Values for Parameter `debug_mode`

9.2.8 Parameter `lr_sc_constraint`

| Set to this value | Description |
|-------------------|--|
| none | Memory access not constrained |
| user1 | Memory access constrained by <code>MEM_CONSTRAINT_USER1</code> |
| user2 | Memory access constrained by <code>MEM_CONSTRAINT_USER2</code> |

Table 9.10: Values for Parameter `lr_sc_constraint`

9.2.9 Parameter `amo_constraint`

| Set to this value | Description |
|-------------------|--|
| none | Memory access not constrained |
| user1 | Memory access constrained by <code>MEM_CONSTRAINT_USER1</code> |
| user2 | Memory access constrained by <code>MEM_CONSTRAINT_USER2</code> |

Table 9.11: Values for Parameter `amo_constraint`

9.2.10 Parameter `Zfinx_version`

| Set to this value | Description |
|-------------------|-----------------------|
| none | Zfinx not implemented |
| 0.4 | Zfinx version 0.4 |
| 0.41 | Zfinx version 0.41 |

Table 9.12: Values for Parameter Zfinx_version

9.2.11 Parameter Zcea_version

| Set to this value | Description |
|-------------------|----------------------|
| none | Zcea not implemented |
| 0.50.1 | Zcea version 0.50.1 |

Table 9.13: Values for Parameter Zceb_version

9.2.12 Parameter Zceb_version

| Set to this value | Description |
|-------------------|----------------------|
| none | Zceb not implemented |
| 0.50.1 | Zceb version 0.50.1 |

Table 9.14: Values for Parameter Zceb_version

9.2.13 Parameter Zcee_version

| Set to this value | Description |
|-------------------|-----------------------|
| none | Zcee not implemented |
| 1.0.0-rc | Zcee version 1.0.0-rc |

Table 9.15: Values for Parameter Zcee_version

9.3 Parameter values

These are the current parameter values.

| Name | Value |
|-------------------------|----------|
| Fundamental | |
| variant | D45 |
| user_version | 2.2 |
| priv_version | 20190608 |
| numHarts | 0 |
| endian | none |
| enable_expanded | F |
| endianFixed | F |
| misa_MXL | 2 |
| misa_Extensions | 0x94312d |
| add_Extensions | |
| sub_Extensions | |
| misa_Extensions_mask | 0x80312d |
| add_Extensions_mask | |
| sub_Extensions_mask | |
| add_implicit_Extensions | |

| | |
|------------------------------|------------------|
| sub_implicit_Extensions | |
| Compressed_Extension | |
| compress_version | legacy |
| Zcea_version | none |
| Zceb_version | none |
| Zcee_version | none |
| Debug_Extension | |
| debug_version | 0.13.2-DRAFT |
| debug_mode | none |
| lr_sc_constraint | user1 |
| amo_constraint | user1 |
| Interrupts_Exceptions | |
| rnmi_version | none |
| mtvec_is_ro | F |
| tvec_align | 0 |
| ecode_mask | 0x7fffffffffffff |
| ecode_nmi | 0 |
| nmi_is_latched | F |
| tval_zero | F |
| tval_zero_ebreak | F |
| tval_ii_code | T |
| trap_preserves_lr | F |
| xret_preserves_lr | F |
| reset_address | 0 |
| nmi_address | 0 |
| CLINT_address | 0 |
| local_int_num | 0 |
| unimp_int_mask | 0 |
| force_mideleg | 0 |
| force_sideleg | 0 |
| no_ideleg | 0 |
| no_edeleg | 0 |
| external_int_id | T |
| Floating_Point | |
| mstatus_fs_mode | write_1 |
| d_requires_f | F |
| enable_fflags_i | F |
| mstatus_FS | 0 |
| Zfh | F |
| Zfhmin | F |
| Zfinx_version | none |
| Simulation_Artifact | |
| use_hw_reg_names | F |
| no_pseudo_inst | F |
| ABI_d | T |

| | |
|---------------------------------|----------------|
| verbose | F |
| traceVolatile | F |
| enable_CSR_bus | F |
| CSR_remap | |
| ASID_cache_size | 8 |
| Memory | |
| updatePTEA | F |
| updatePTED | F |
| unaligned_low_pri | F |
| unaligned | F |
| unalignedAMO | F |
| amo_aborts_lr_sc | F |
| ASID_bits | 0 |
| lr_sc_grain | 1 |
| Sv_modes | 0x701 |
| Instruction_CSR_Behavior | |
| wfi_is_nop | F |
| counteren_mask | 127 |
| noinhibit_mask | 0 |
| cycle_undefined | F |
| mcycle_undefined | F |
| time_undefined | F |
| instret_undefined | F |
| minstret_undefined | F |
| hpmcounter_undefined | F |
| mhpmcounter_undefined | F |
| CSR_Masks | |
| mtvec_mask | 0 |
| stvec_mask | 0 |
| utvec_mask | 0 |
| tdata1_mask | 0xffffffffffff |
| mip_mask | 0x337 |
| sip_mask | 0x103 |
| uip_mask | 1 |
| mtvec_sext | F |
| stvec_sext | F |
| utvec_sext | F |
| MXL_writable | F |
| SXL_writable | F |
| UXL_writable | F |
| Trigger | |
| tinfo_undefined | F |
| tcontrol_undefined | F |
| mcontext_undefined | F |
| scontext_undefined | F |

| | |
|--------------------------|------------|
| mscontext_undefined | F |
| amo_trigger | F |
| no_hit | F |
| no_sselect_2 | F |
| trigger_num | 4 |
| tinfo | 61 |
| mcontext_bits | 13 |
| scontext_bits | 34 |
| mvalue_bits | 13 |
| svalue_bits | 34 |
| mcontrol_maskmax | 63 |
| PMP Configuration | |
| PMP_grain | 0 |
| PMP_registers | 0 |
| PMP_max_page | 0 |
| PMP_decompose | F |
| PMP_undefined | F |
| PMP_maskparams | F |
| PMP_initialparams | F |
| Other Extensions | |
| Svnapot_page_mask | 0 |
| Smstateen | F |
| Svpbmt | F |
| Svinal | F |
| Zicsr | T |
| Zifencei | T |
| Zicbom | F |
| Zicbop | F |
| Zicboz | F |
| Zmmul | F |
| CSR Defaults | |
| mvendorid | 0x31e |
| marchid | 0x80000045 |
| mimpid | 16 |
| mhartid | 0 |
| mtvec | 0 |
| Fast Interrupt | |
| CLICLEVELS | 0 |
| andesExtensions | |
| PMA_grain* | 0 |
| milmb* | 0 |
| mdlmb* | 0 |
| mmsc_cfg* | 0x2003b039 |
| micm_cfg* | 0x51acb |
| mdcm_cfg* | 0x51ada |

| | |
|----------------|----|
| mvec_cfg* | 0 |
| uitb* | 0 |
| ml2c_ctl_base* | 0 |
| milmbMask* | 14 |
| mdlmbMask* | 14 |
| aceFile* | |
| aceLibrary* | |
| aceDiagLevel* | 0 |

Table 9.16: Parameter values

* Parameters marked with an asterisk are part of the processor extension library.

Chapter 10

Execution Modes

| Mode | Code | Description |
|-------------|------|-----------------|
| User | 0 | User mode |
| Supervisor | 1 | Supervisor mode |
| Machine | 3 | Machine mode |

Table 10.1: Modes implemented in: Hart

Chapter 11

Exceptions

| Exception | Code | Description |
|------------------------------|------------|--|
| InstructionAddressMisaligned | 0 | Fetch from unaligned address |
| InstructionAccessFault | 1 | No access permission for fetch |
| IllegalInstruction | 2 | Undecoded, unimplemented or disabled instruction |
| Breakpoint | 3 | EBREAK instruction executed |
| LoadAddressMisaligned | 4 | Load from unaligned address |
| LoadAccessFault | 5 | No access permission for load |
| StoreAMOAddressMisaligned | 6 | Store/atomic memory operation at unaligned address |
| StoreAMOAccessFault | 7 | No access permission for store/atomic memory operation |
| EnvironmentCallFromUMode | 8 | ECALL instruction executed in User mode |
| EnvironmentCallFromSMode | 9 | ECALL instruction executed in Supervisor mode |
| EnvironmentCallFromMMode | 11 | ECALL instruction executed in Machine mode |
| InstructionPageFault | 12 | Page fault at fetch address |
| LoadPageFault | 13 | Page fault at load address |
| StoreAMOPageFault | 15 | Page fault at store/atomic memory operation address |
| HSP_OVF | 32 | Stack overflow |
| HSP_UDF | 33 | Stack underflow |
| USWInterrupt | 64 | User software interrupt |
| SSWInterrupt | 65 | Supervisor software interrupt |
| MSWInterrupt | 67 | Machine software interrupt |
| UTimerInterrupt | 68 | User timer interrupt |
| STimerInterrupt | 69 | Supervisor timer interrupt |
| MTimerInterrupt | 71 | Machine timer interrupt |
| UExternalInterrupt | 72 | User external interrupt |
| SExternalInterrupt | 73 | Supervisor external interrupt |
| MExternalInterrupt | 75 | Machine external interrupt |
| GenericNMI | 4294967295 | Generic NMI |

Table 11.1: Exceptions implemented in: Hart

Chapter 12

Hierarchy of the model

A CPU core may be configured to instance many processors of a Symmetrical Multi Processor (SMP). A CPU core may also have sub elements within a processor, for example hardware threading blocks.

OVP processor models can be written to include SMP blocks and to have many levels of hierarchy. Some OVP CPU models may have a fixed hierarchy, and some may be configured by settings in a configuration register. Please see the register definitions of this model.

This model documentation shows the settings and hierarchy of the default settings for this model variant.

12.1 Level 1: Hart

This level in the model hierarchy has 6 commands.

This level in the model hierarchy has 6 register groups:

| Group name | Registers |
|-------------------------------|-----------|
| Core | 33 |
| Floating_point | 32 |
| User_Control_and_Status | 46 |
| Supervisor_Control_and_Status | 22 |
| Machine_Control_and_Status | 125 |
| Integration_support | 6 |

Table 12.1: Register groups

This level in the model hierarchy has no children.

Chapter 13

Model Commands

A Processor model can implement one or more **Model Commands** available to be invoked from the simulator command line, from the OP API or from the Imperas Multiprocessor Debugger.

13.1 Level 1: Hart

13.1.1 debugflags

show or modify the processor debug flags

| Argument | Type | Description |
|----------|---------|---|
| -get | Boolean | print current processor flags value |
| -mask | Boolean | print valid debug flag bits |
| -set | Int32 | new processor flags (only flags 0x00000006 can be modified) |

Table 13.1: debugflags command arguments

13.1.2 dumpTLB

13.1.2.1 Argument description

show TLB contents

13.1.3 getCSRIndex

Return index for a named CSR (or -1 if no matching CSR)

| Argument | Type | Description |
|----------|--------|-------------|
| -name | String | CSR name |

Table 13.2: getCSRIndex command arguments

13.1.4 isync

specify instruction address range for synchronous execution

| Argument | Type | Description |
|----------|------|-------------|
|----------|------|-------------|

| | | |
|------------|-------|--|
| -addresshi | Uns64 | end address of synchronous execution range |
| -addresslo | Uns64 | start address of synchronous execution range |

Table 13.3: isync command arguments

13.1.5 itrace

enable or disable instruction tracing

| Argument | Type | Description |
|-------------------|---------|---|
| -after | Uns64 | apply after this many instructions |
| -enable | Boolean | enable instruction tracing |
| -instructioncount | Boolean | include the instruction number in each trace |
| -memory | String | show memory accesses by this instruction. Argument can be any combination of X (execute), A (load or store access) and S (system) |
| -mode | Boolean | show processor mode changes |
| -off | Boolean | disable instruction tracing |
| -on | Boolean | enable instruction tracing |
| -processorname | Boolean | Include processor name in all trace lines |
| -registerchange | Boolean | show registers changed by this instruction |
| -registers | Boolean | show registers after each trace |

Table 13.4: itrace command arguments

13.1.6 listCSRs

13.1.6.1 Argument description

List all CSRs in index order

Chapter 14

Registers

14.1 Level 1: Hart

14.1.1 Core

Registers at level:1, type:Hart group:Core

| Name | Bits | Initial-Hex | RW | Description |
|------|------|-------------|----|-----------------|
| zero | 64 | 0 | r- | |
| ra | 64 | 0 | rw | |
| sp | 64 | 0 | rw | stack pointer |
| gp | 64 | 0 | rw | |
| tp | 64 | 0 | rw | |
| t0 | 64 | 0 | rw | |
| t1 | 64 | 0 | rw | |
| t2 | 64 | 0 | rw | |
| s0 | 64 | 0 | rw | |
| s1 | 64 | 0 | rw | |
| a0 | 64 | 0 | rw | |
| a1 | 64 | 0 | rw | |
| a2 | 64 | 0 | rw | |
| a3 | 64 | 0 | rw | |
| a4 | 64 | 0 | rw | |
| a5 | 64 | 0 | rw | |
| a6 | 64 | 0 | rw | |
| a7 | 64 | 0 | rw | |
| s2 | 64 | 0 | rw | |
| s3 | 64 | 0 | rw | |
| s4 | 64 | 0 | rw | |
| s5 | 64 | 0 | rw | |
| s6 | 64 | 0 | rw | |
| s7 | 64 | 0 | rw | |
| s8 | 64 | 0 | rw | |
| s9 | 64 | 0 | rw | |
| s10 | 64 | 0 | rw | |
| s11 | 64 | 0 | rw | |
| t3 | 64 | 0 | rw | |
| t4 | 64 | 0 | rw | |
| t5 | 64 | 0 | rw | |
| t6 | 64 | 0 | rw | |
| pc | 64 | 0 | rw | program counter |

Table 14.1: Registers at level 1, type:Hart group:Core

14.1.2 Floating_point

Registers at level:1, type:Hart group:Floating_point

| Name | Bits | Initial-Hex | RW | Description |
|------|------|-------------|----|-------------|
| ft0 | 64 | 0 | rw | |
| ft1 | 64 | 0 | rw | |
| ft2 | 64 | 0 | rw | |
| ft3 | 64 | 0 | rw | |
| ft4 | 64 | 0 | rw | |
| ft5 | 64 | 0 | rw | |
| ft6 | 64 | 0 | rw | |
| ft7 | 64 | 0 | rw | |
| fs0 | 64 | 0 | rw | |
| fs1 | 64 | 0 | rw | |
| fa0 | 64 | 0 | rw | |
| fa1 | 64 | 0 | rw | |
| fa2 | 64 | 0 | rw | |
| fa3 | 64 | 0 | rw | |
| fa4 | 64 | 0 | rw | |
| fa5 | 64 | 0 | rw | |
| fa6 | 64 | 0 | rw | |
| fa7 | 64 | 0 | rw | |
| fs2 | 64 | 0 | rw | |
| fs3 | 64 | 0 | rw | |
| fs4 | 64 | 0 | rw | |
| fs5 | 64 | 0 | rw | |
| fs6 | 64 | 0 | rw | |
| fs7 | 64 | 0 | rw | |
| fs8 | 64 | 0 | rw | |
| fs9 | 64 | 0 | rw | |
| fs10 | 64 | 0 | rw | |
| fs11 | 64 | 0 | rw | |
| ft8 | 64 | 0 | rw | |
| ft9 | 64 | 0 | rw | |
| ft10 | 64 | 0 | rw | |
| ft11 | 64 | 0 | rw | |

Table 14.2: Registers at level 1, type:Hart group:Floating_point

14.1.3 User_Control_and_Status

Registers at level:1, type:Hart group:User_Control_and_Status

| Name | Bits | Initial-Hex | RW | Description |
|----------|------|-------------|----|-----------------------------------|
| ustatus | 64 | 0 | rw | User Status |
| fflags | 64 | 0 | rw | Floating-Point Flags |
| frm | 64 | 0 | rw | Floating-Point Rounding Mode |
| fcsr | 64 | 0 | rw | Floating-Point Control and Status |
| uie | 64 | 0 | rw | User Interrupt Enable |
| utvec | 64 | 0 | rw | User Trap-Vector Base-Address |
| uscratch | 64 | 0 | rw | User Scratch |

| | | | | |
|-----------------|----|---|----|---|
| uepc | 64 | 0 | rw | User Exception Program Counter |
| ucause | 64 | 0 | rw | User Cause |
| utval | 64 | 0 | rw | User Trap Value |
| uip | 64 | 0 | rw | User Interrupt Pending |
| uitb* | 64 | 0 | rw | Instruction Table Base Address |
| ucctlbeginaddr* | 64 | 0 | rw | User CCTL Begin Address (register only) |
| ucctlcommand* | 64 | 0 | rw | User CCTL Command (register only) |
| cycle* | 64 | 0 | rw | Cycle Counter |
| time* | 64 | 0 | r- | Timer |
| instret* | 64 | 0 | rw | Instructions Retired |
| hpmcounter3* | 64 | 0 | rw | Performance Monitor Counter |
| hpmcounter4* | 64 | 0 | rw | Performance Monitor Counter |
| hpmcounter5* | 64 | 0 | rw | Performance Monitor Counter |
| hpmcounter6* | 64 | 0 | rw | Performance Monitor Counter |
| hpmcounter7 | 64 | 0 | r- | Performance Monitor Counter 7 |
| hpmcounter8 | 64 | 0 | r- | Performance Monitor Counter 8 |
| hpmcounter9 | 64 | 0 | r- | Performance Monitor Counter 9 |
| hpmcounter10 | 64 | 0 | r- | Performance Monitor Counter 10 |
| hpmcounter11 | 64 | 0 | r- | Performance Monitor Counter 11 |
| hpmcounter12 | 64 | 0 | r- | Performance Monitor Counter 12 |
| hpmcounter13 | 64 | 0 | r- | Performance Monitor Counter 13 |
| hpmcounter14 | 64 | 0 | r- | Performance Monitor Counter 14 |
| hpmcounter15 | 64 | 0 | r- | Performance Monitor Counter 15 |
| hpmcounter16 | 64 | 0 | r- | Performance Monitor Counter 16 |
| hpmcounter17 | 64 | 0 | r- | Performance Monitor Counter 17 |
| hpmcounter18 | 64 | 0 | r- | Performance Monitor Counter 18 |
| hpmcounter19 | 64 | 0 | r- | Performance Monitor Counter 19 |
| hpmcounter20 | 64 | 0 | r- | Performance Monitor Counter 20 |
| hpmcounter21 | 64 | 0 | r- | Performance Monitor Counter 21 |
| hpmcounter22 | 64 | 0 | r- | Performance Monitor Counter 22 |
| hpmcounter23 | 64 | 0 | r- | Performance Monitor Counter 23 |
| hpmcounter24 | 64 | 0 | r- | Performance Monitor Counter 24 |
| hpmcounter25 | 64 | 0 | r- | Performance Monitor Counter 25 |
| hpmcounter26 | 64 | 0 | r- | Performance Monitor Counter 26 |
| hpmcounter27 | 64 | 0 | r- | Performance Monitor Counter 27 |
| hpmcounter28 | 64 | 0 | r- | Performance Monitor Counter 28 |
| hpmcounter29 | 64 | 0 | r- | Performance Monitor Counter 29 |
| hpmcounter30 | 64 | 0 | r- | Performance Monitor Counter 30 |
| hpmcounter31 | 64 | 0 | r- | Performance Monitor Counter 31 |

Table 14.3: Registers at level 1, type:Hart group:User_Control_and_Status

* Registers marked with an asterisk are part of the processor extension library.

14.1.4 Supervisor_Control_and_Status

Registers at level:1, type:Hart group:Supervisor_Control_and_Status

| Name | Bits | Initial-Hex | RW | Description |
|------------|------|-------------|----|-------------------------------------|
| sstatus | 64 | 2 00000000 | rw | Supervisor Status |
| sedeleg | 64 | 0 | rw | Supervisor Exception Delegation |
| sideleg | 64 | 0 | rw | Supervisor Interrupt Delegation |
| sie | 64 | 0 | rw | Supervisor Interrupt Enable |
| stvec | 64 | 0 | rw | Supervisor Trap-Vector Base-Address |
| scounteren | 64 | 0 | rw | Supervisor Counter Enable |

| | | | | |
|-----------------|----|---|----|---|
| sscratch | 64 | 0 | rw | Supervisor Scratch |
| sepc | 64 | 0 | rw | Supervisor Exception Program Counter |
| scause | 64 | 0 | rw | Supervisor Cause |
| stval | 64 | 0 | rw | Supervisor Trap Value |
| sip | 64 | 0 | rw | Supervisor Interrupt Pending |
| satp | 64 | 0 | rw | Supervisor Address Translation and Protection |
| scontext | 64 | 0 | rw | Trigger Supervisor Context |
| scounterinten* | 64 | 0 | rw | Supervisor Counter Interrupt Enable |
| scountermask_m* | 64 | 0 | rw | Supervisor Counter Mask for Machine Mode |
| scountermask_s* | 64 | 0 | rw | Supervisor Counter Mask for Supervisor Mode |
| scountermask_u* | 64 | 0 | rw | Supervisor Counter Mask for User Mode |
| scounterovf* | 64 | 0 | rw | Supervisor Counter Overflow Status |
| shpmevent3* | 64 | 0 | rw | Supervisor Performance Monitor Event Select |
| shpmevent4* | 64 | 0 | rw | Supervisor Performance Monitor Event Select |
| shpmevent5* | 64 | 0 | rw | Supervisor Performance Monitor Event Select |
| shpmevent6* | 64 | 0 | rw | Supervisor Performance Monitor Event Select |

Table 14.4: Registers at level 1, type:Hart group:Supervisor_Control_and_Status

* Registers marked with an asterisk are part of the processor extension library.

14.1.5 Machine_Control_and_Status

Registers at level:1, type:Hart group:Machine_Control_and_Status

| Name | Bits | Initial-Hex | RW | Description |
|---------------|------|----------------------|----|---|
| mstatus | 64 | a 00001800 | rw | Machine Status |
| misa | 64 | 80000000 0094312d | rw | ISA and Extensions |
| medeleg | 64 | 0 | rw | Machine Exception Delegation |
| mideleg | 64 | 0 | rw | Machine Interrupt Delegation |
| mie | 64 | 0 | rw | Machine Interrupt Enable |
| mtvec | 64 | 0 | rw | Machine Trap-Vector Base-Address |
| mcounteren | 64 | 0 | rw | Machine Counter Enable |
| mcountinhibit | 64 | 0 | rw | Machine Counter Inhibit |
| mhpmevent3* | 64 | 0 | rw | Machine Performance Monitor Event Select |
| mhpmevent4* | 64 | 0 | rw | Machine Performance Monitor Event Select |
| mhpmevent5* | 64 | 0 | rw | Machine Performance Monitor Event Select |
| mhpmevent6* | 64 | 0 | rw | Machine Performance Monitor Event Select |
| mhpmevent7 | 64 | 0 | rw | Machine Performance Monitor Event Select 7 |
| mhpmevent8 | 64 | 0 | rw | Machine Performance Monitor Event Select 8 |
| mhpmevent9 | 64 | 0 | rw | Machine Performance Monitor Event Select 9 |
| mhpmevent10 | 64 | 0 | rw | Machine Performance Monitor Event Select 10 |
| mhpmevent11 | 64 | 0 | rw | Machine Performance Monitor Event Select 11 |
| mhpmevent12 | 64 | 0 | rw | Machine Performance Monitor Event Select 12 |
| mhpmevent13 | 64 | 0 | rw | Machine Performance Monitor Event Select 13 |
| mhpmevent14 | 64 | 0 | rw | Machine Performance Monitor Event Select 14 |
| mhpmevent15 | 64 | 0 | rw | Machine Performance Monitor Event Select 15 |
| mhpmevent16 | 64 | 0 | rw | Machine Performance Monitor Event Select 16 |
| mhpmevent17 | 64 | 0 | rw | Machine Performance Monitor Event Select 17 |
| mhpmevent18 | 64 | 0 | rw | Machine Performance Monitor Event Select 18 |
| mhpmevent19 | 64 | 0 | rw | Machine Performance Monitor Event Select 19 |
| mhpmevent20 | 64 | 0 | rw | Machine Performance Monitor Event Select 20 |
| mhpmevent21 | 64 | 0 | rw | Machine Performance Monitor Event Select 21 |
| mhpmevent22 | 64 | 0 | rw | Machine Performance Monitor Event Select 22 |

| | | | | |
|-----------------|----|-----------------|----|---|
| mhpmevent23 | 64 | 0 | rw | Machine Performance Monitor Event Select 23 |
| mhpmevent24 | 64 | 0 | rw | Machine Performance Monitor Event Select 24 |
| mhpmevent25 | 64 | 0 | rw | Machine Performance Monitor Event Select 25 |
| mhpmevent26 | 64 | 0 | rw | Machine Performance Monitor Event Select 26 |
| mhpmevent27 | 64 | 0 | rw | Machine Performance Monitor Event Select 27 |
| mhpmevent28 | 64 | 0 | rw | Machine Performance Monitor Event Select 28 |
| mhpmevent29 | 64 | 0 | rw | Machine Performance Monitor Event Select 29 |
| mhpmevent30 | 64 | 0 | rw | Machine Performance Monitor Event Select 30 |
| mhpmevent31 | 64 | 0 | rw | Machine Performance Monitor Event Select 31 |
| mscratch | 64 | 0 | rw | Machine Scratch |
| mepc | 64 | 0 | rw | Machine Exception Program Counter |
| mcause | 64 | 0 | rw | Machine Cause |
| mtval | 64 | 0 | rw | Machine Trap Value |
| mip | 64 | 0 | rw | Machine Interrupt Pending |
| pmpcfg0 | 64 | 0 | rw | Physical Memory Protection Configuration 0 |
| pmpcfg2 | 64 | 0 | rw | Physical Memory Protection Configuration 2 |
| pmpaddr0 | 64 | 0 | rw | Physical Memory Protection Address 0 |
| pmpaddr1 | 64 | 0 | rw | Physical Memory Protection Address 1 |
| pmpaddr2 | 64 | 0 | rw | Physical Memory Protection Address 2 |
| pmpaddr3 | 64 | 0 | rw | Physical Memory Protection Address 3 |
| pmpaddr4 | 64 | 0 | rw | Physical Memory Protection Address 4 |
| pmpaddr5 | 64 | 0 | rw | Physical Memory Protection Address 5 |
| pmpaddr6 | 64 | 0 | rw | Physical Memory Protection Address 6 |
| pmpaddr7 | 64 | 0 | rw | Physical Memory Protection Address 7 |
| pmpaddr8 | 64 | 0 | rw | Physical Memory Protection Address 8 |
| pmpaddr9 | 64 | 0 | rw | Physical Memory Protection Address 9 |
| pmpaddr10 | 64 | 0 | rw | Physical Memory Protection Address 10 |
| pmpaddr11 | 64 | 0 | rw | Physical Memory Protection Address 11 |
| pmpaddr12 | 64 | 0 | rw | Physical Memory Protection Address 12 |
| pmpaddr13 | 64 | 0 | rw | Physical Memory Protection Address 13 |
| pmpaddr14 | 64 | 0 | rw | Physical Memory Protection Address 14 |
| pmpaddr15 | 64 | 0 | rw | Physical Memory Protection Address 15 |
| tselect | 64 | 0 | rw | Trigger Register Select |
| tdata1 | 64 | 0 | rw | Trigger Data 1 |
| tdata2 | 64 | 0 | rw | Trigger Data 2 |
| tdata3 | 64 | 0 | rw | Trigger Data 3 |
| tinfo | 64 | 3d | rw | Trigger Info |
| tcontrol | 64 | 0 | rw | Trigger Control |
| mcontext | 64 | 0 | rw | Trigger Machine Context |
| milmb* | 64 | 0 | rw | Instruction Local Memory Base |
| mdlmb* | 64 | 0 | rw | Data Local Memory Base |
| mnvec* | 64 | 0 | rw | NMI Vector Base Address |
| mxstatus* | 64 | 0 | rw | Machine Extended Status |
| mpft_ctl* | 64 | 0 | rw | Performance Throttling Control |
| mhsp_ctl* | 64 | 0 | rw | Machine Hardware Stack Protection Control |
| mssp_bound* | 64 | ffffff ffffffff | rw | Machine SP Bound |
| mssp_base* | 64 | ffffff ffffffff | rw | Machine SP Base |
| mdcause* | 64 | 0 | rw | Machine Detailed Trap Cause |
| mcache_ctl* | 64 | 0 | rw | Cache Control |
| mcctlbeginaddr* | 64 | 0 | rw | Machine CCTL Begin Address (register only) |
| mcctlcommand* | 64 | 0 | rw | Machine CCTL Command (register only) |
| mcctldata* | 64 | 0 | rw | Machine CCTL Data (register only) |
| mcounterwen* | 64 | 0 | rw | Machine Counter Write Enable |
| mcounterinten* | 64 | 0 | rw | Machine Counter Interrupt Enable |
| mmisc_ctl* | 64 | 0 | rw | Machine Miscellaneous Control |
| mcountermask_m* | 64 | 0 | rw | Machine Counter Mask for Machine Mode |

| | | | | |
|-----------------|----|----------|----|--|
| mcountermask_s* | 64 | 0 | rw | Machine Counter Mask for Supervisor Mode |
| mcountermask_u* | 64 | 0 | rw | Machine Counter Mask for User Mode |
| mcounterovf* | 64 | 0 | rw | Machine Counter Overflow Status |
| mcycle* | 64 | 0 | rw | Machine Cycle Counter |
| minstret* | 64 | 0 | rw | Machine Instructions Retired |
| mhpmcounter3* | 64 | 0 | rw | Machine Performance Monitor Counter |
| mhpmcounter4* | 64 | 0 | rw | Machine Performance Monitor Counter |
| mhpmcounter5* | 64 | 0 | rw | Machine Performance Monitor Counter |
| mhpmcounter6* | 64 | 0 | rw | Machine Performance Monitor Counter |
| mhpmcounter7 | 64 | 0 | rw | Machine Performance Monitor Counter 7 |
| mhpmcounter8 | 64 | 0 | rw | Machine Performance Monitor Counter 8 |
| mhpmcounter9 | 64 | 0 | rw | Machine Performance Monitor Counter 9 |
| mhpmcounter10 | 64 | 0 | rw | Machine Performance Monitor Counter 10 |
| mhpmcounter11 | 64 | 0 | rw | Machine Performance Monitor Counter 11 |
| mhpmcounter12 | 64 | 0 | rw | Machine Performance Monitor Counter 12 |
| mhpmcounter13 | 64 | 0 | rw | Machine Performance Monitor Counter 13 |
| mhpmcounter14 | 64 | 0 | rw | Machine Performance Monitor Counter 14 |
| mhpmcounter15 | 64 | 0 | rw | Machine Performance Monitor Counter 15 |
| mhpmcounter16 | 64 | 0 | rw | Machine Performance Monitor Counter 16 |
| mhpmcounter17 | 64 | 0 | rw | Machine Performance Monitor Counter 17 |
| mhpmcounter18 | 64 | 0 | rw | Machine Performance Monitor Counter 18 |
| mhpmcounter19 | 64 | 0 | rw | Machine Performance Monitor Counter 19 |
| mhpmcounter20 | 64 | 0 | rw | Machine Performance Monitor Counter 20 |
| mhpmcounter21 | 64 | 0 | rw | Machine Performance Monitor Counter 21 |
| mhpmcounter22 | 64 | 0 | rw | Machine Performance Monitor Counter 22 |
| mhpmcounter23 | 64 | 0 | rw | Machine Performance Monitor Counter 23 |
| mhpmcounter24 | 64 | 0 | rw | Machine Performance Monitor Counter 24 |
| mhpmcounter25 | 64 | 0 | rw | Machine Performance Monitor Counter 25 |
| mhpmcounter26 | 64 | 0 | rw | Machine Performance Monitor Counter 26 |
| mhpmcounter27 | 64 | 0 | rw | Machine Performance Monitor Counter 27 |
| mhpmcounter28 | 64 | 0 | rw | Machine Performance Monitor Counter 28 |
| mhpmcounter29 | 64 | 0 | rw | Machine Performance Monitor Counter 29 |
| mhpmcounter30 | 64 | 0 | rw | Machine Performance Monitor Counter 30 |
| mhpmcounter31 | 64 | 0 | rw | Machine Performance Monitor Counter 31 |
| mvendorid | 64 | 31e | r- | Vendor ID |
| marchid | 64 | 80000045 | r- | Architecture ID |
| mimpid | 64 | 10 | r- | Implementation ID |
| mhartid | 64 | 0 | r- | Hardware Thread ID |
| micm_cfg* | 64 | 51acb | r- | Instruction Cache/Memory Configuration |
| mddm_cfg* | 64 | 51ada | r- | Data Cache/Memory Configuration |
| mmisc_cfg* | 64 | 2003b039 | r- | Miscellaneous Configuration |

Table 14.5: Registers at level 1, type:Hart group:Machine_Control_and_Status

* Registers marked with an asterisk are part of the processor extension library.

14.1.6 Integration support

Registers at level:1, type:Hart group:Integration_support

| Name | Bits | Initial-Hex | RW | Description |
|--------------|------|-----------------|----|---|
| LRSCAddress | 64 | ffffff ffffffff | rw | LR/SC active lock address |
| commercial | 8 | 0 | r- | Commercial feature in use |
| PTWStage | 8 | 0 | r- | PTW active stage (0:none 1:HS 2:VS 3:G) |
| PTWInputAddr | 64 | 0 | r- | PTW input address |

| | | | | |
|----------|---|---|----|--------------------------------------|
| PTWLevel | 8 | 0 | r- | PTW active level |
| ASYNCPPE | 8 | 0 | r- | Asynchronous Event Pending & Enabled |

Table 14.6: Registers at level 1, type:Hart group:Integration_support